Test Framework

Requirements

A Web Based System to Support Testing Multiple Program Modules

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

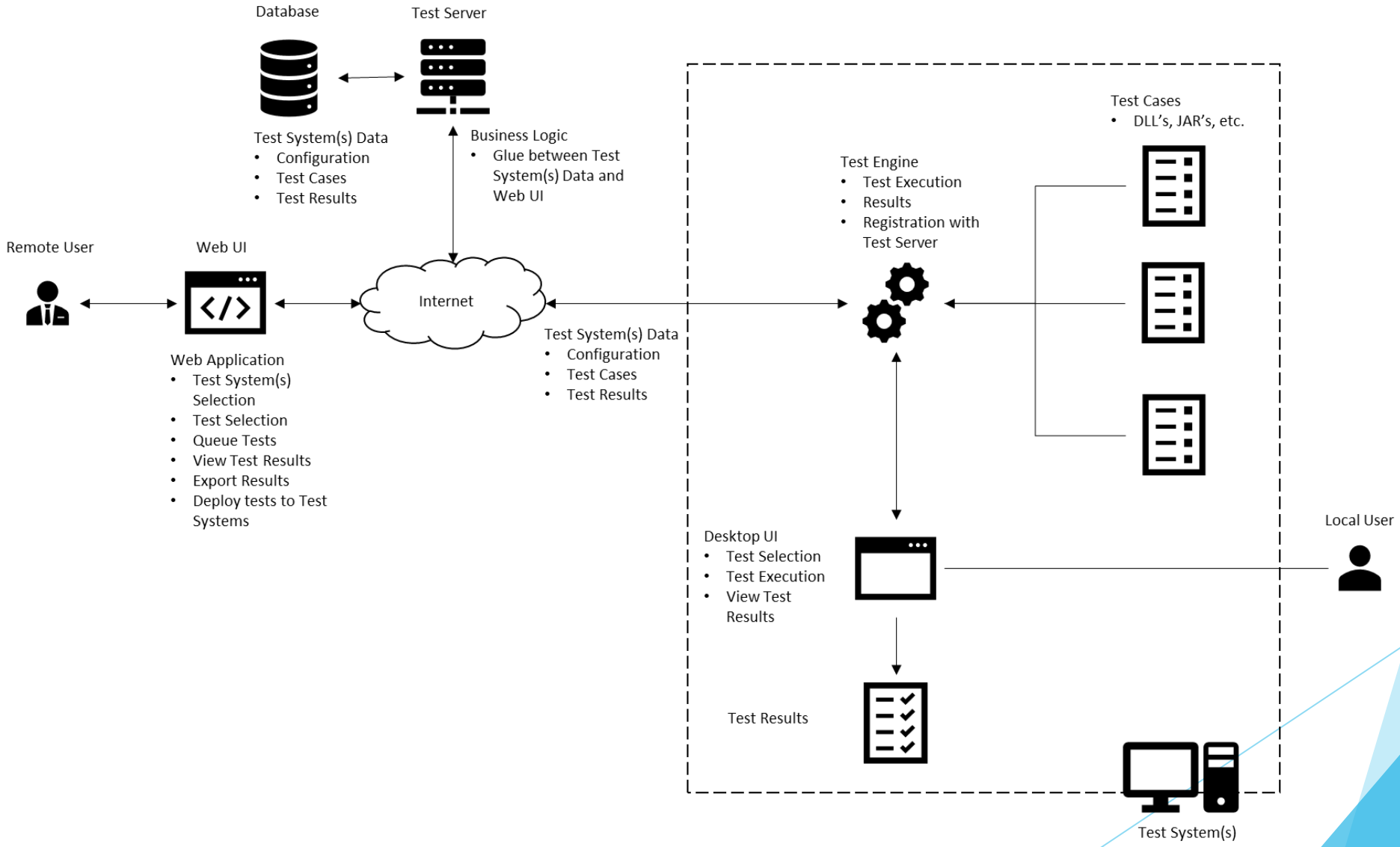
Preface

- Developing large scale software with complex features, interactions, or complex system interfaces (APIs) is best built and tested incrementally.
- Build a basic core, with a small number of packages, then add features one-at-a-time with new packages or new program code to existing packages.
- Each time new functionality is added, the application is built and tested.
- The Test Framework Application will allow the development team(s), and other users of the system to define tests which run with exception handling and results logging.
- Goal of the Test Framework is to allow program testing with out proliferating code with many try-catch blocks, debug statements, assertions, or verbose logging.
- The Test Framework will be easy to use and accessed by the user's web browser. The system itself being cloud hosted in multiple regions so performance is always top notch.

Introduction

- Web based, cloud hosted solution.
- Ability to run locally; i.e. sans web / cloud hosted.
- Can test multiple program modules or blocks of program code simultaneously.
- Multiple methods of access or delivery (dynamic link library, XML, JSON, etc.).
- Support for Multiple Languages (C++, C#, Java, Python).
- Scalable solution.
- Cloud hosted in multiple regions for best performance, reduction in latency, support high availability and disaster recovery of solution.

System Overview



System Overview

The key components are:

- Users
  - Local User
  - Remote User
- Web User Interface
- Test Server
  - Test Server
  - Database
- Test System
  - Test Engine
  - Test Cases
  - Desktop User Interface
  - Test Results

System Users

There are nine types of users of this system. They are:

- Software Developers. These users are the developers of the software. They create designs for new features, build and run tests for each new feature.
- 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.
- 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.
- 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.
- 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.
- System Engineers. These users are the technical leads. In some organizations they are considered the architects of the system.
- 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.
- IT Managers (Mostly line level managers).
- System Administrators. These users manage and maintain the system, accessibility, and perform system maintenance upgrades.

User Roles and Accessibility

- **Local User** – user with locally installed test engine on PC/laptop. Able to register with Test Server to register test engine in database. Able to run tests locally. Uses UI with internal engine.
- **Remote User** – user with same capability as local user, but has ability to view available test engines across the web and to use other hardware, servers, infrastructure for test purposes as well as capability to view archived results.
- **Administrator** – user with the ability to install application remotely, update application services, add remote users, de-register test engines, perform HA/DR testing.

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 shall have:

- Graphic User Interface (GUI).
- Web enabled front end.
- Capability to run multiple tests simultaneously.
- Capability to ensure that no one test can tie up system resources.
- Ability to allow multiple users to use the system at the same time.
- 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.

Requirements (User and System Level)

- 2.1 The users of this Test Framework system shall have the ability to setup and run individual unit tests of program code, hardware, and infrastructure, as well as run multiple tests simultaneously. They need to be able to stress performance, ensure scalability, and diagnose system interface issues. Test results shall be saved for future recall, and configuration of the system maintained and stored.

- 2.1.1 The test engine shall not require changing and recompiling the program each time a test is run. [0]
- 2.1.2 The system shall be implemented as a client-server system wherein the Web User Interface (WebUI) communicates with the test engine and test server over the internet. [1]
- 2.1.3 The system shall employ a database in which to store all test data, including configuration data, test cases, and test results. [1]
- 2.2.4 The system shall employ a test server, which will provide the business logic and serve as the API between the database and the WebUI. [1]
- 2.1.5 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, managed services where needed. [2]
- 2.1.6 The system shall allow the ability to create additional environments for specialized testing upon demand. [2]

Requirements (User and System Level)

- 2.2 The user shall have the ability to view the results of the test as each test completes. He/she shall also have the ability to view the log files where all results and relevant output are stored. The logs should contain various levels of information depending on the specified log level and shall display a time and date stamp.

- 2.2.1 The system shall display whether each test failed or succeeded. [0]
- 2.2.2 The application shall log all test results to the output file specified in the GUI. [0]
- 2.2.3 The log component shall show different levels of logging (INFO, DEBUG, ERROR). [1]
  - 2.2.3.1 INFO shall describe specific information for test pass/fail reporting.
  - 2.2.3.2 DEBUG shall describe information provided by the programmer/developer to aid in debugging the test.
  - 2.2.3.3 ERROR shall describe the most detailed debugging output for examination of software test failures.
- 2.2.4 The log shall display the time and date stamp for each test. [1]
- 2.2.5 The log shall display the duration of each test. [1]

Requirements (User and System Level)

- 2.3 Users shall have the ability to provide a test case where several tests are sent in quick succession to demonstrate the application executes tests concurrently.

- 2.3.1 The test case shall consist of several tests of varying duration that can run simultaneously. [0]
- 2.3.2 Upon completion, the system shall post a ready status message and await the next test. [0]
- 2.3.3 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). [0]
  - 2.3.3.1 The system shall allow specification of the thread pool size. [1]
  - 2.3.3.2 The starting default minimum thread count shall be 5. [1]
  - 2.3.3.3 The starting default maximum thread count shall be 15 (this keeps the application from spawning too many threads). [1]

Requirements (User and System Level)

- 2.4 The test engine shall run on Windows platform and should run on Linux and Mac platforms. User access to the system shall be provided through a Web User Interface (WebUI). The WebUI shall support the Firefox web browser and should support Google Chrome and Microsoft Edge.

- 2.4.1 The test engine shall be an application that can run on the Windows platform. [0]
- 2.4.2 The test engine should run on Linux and Mac platforms. [1]
- 2.4.3 Client access to the system shall be provided through a WebUI to be accessed via a standard web browser. [1]
  - 2.4.3.1 The system shall support the Firefox web browser. [1]
  - 2.4.3.2 The system should support the Microsoft Edge and Google Chrome web browsers. [2]

Requirements (User and System Level)

- 2.5 The system shall handle application failures and errors as well as test failures and errors.

- 2.5.1 The system shall handle exceptions thrown by the application during testing with clear user output. [0]

Requirements (User and System Level)

- 2.6 The system shall be user friendly, have a graphic user interface, and use a web-enabled client (browser). The system shall be installable locally on the user's machine and use its own desktop interface.

- 2.6.1 The system shall have a desktop interface for locally installed users. [0]
- 2.6.2 The system shall have a WebUI for remote users. [1]
- 2.6.3 The system shall be available on demand remotely via the WebUI. [1]
- 2.6.4 The desktop interface shall:
  - 2.6.4.1 Display all possible tests and allow the user to select all tests they wish to run. [0]
  - 2.6.4.2 Display the selected list of all tests to be run (container object on GUI). [0]
  - 2.6.4.3 Show test progress and status on the GUI. [0]
  - 2.6.4.4 Display the results of each test in real-time. [1]
  - 2.6.4.5 Allow the user to specify an output file in which to log the test results. [1]

Requirements (User and System Level) 2.6 Continued

- 2.6.5 The WebUI shall:
  - 2.6.5.1 Display all possible tests and allow the user to select all tests they wish to run. [1]
  - 2.6.5.2 Display the selected list of all tests to be run (container object on GUI). [1]
  - 2.6.5.3 Show test progress and status on the GUI. [1]
  - 2.6.5.4 Display the results of each test in real-time. [2]
  - 2.6.5.5 Allow the user to specify an output file in which to log the test results. [2]
  - 2.6.5.6 Execute tests on available clients. [1]
  - 2.6.5.7 Export current and prior test results for specific clients. [2]

Requirements (User and System Level)

- 2.7 The user shall log into the system via the WebUI and the system will authenticate the user. Once logged into the system and user role determination has been made, the user can register/de-register his/her machine with the Test Server.

- 2.7.1 The Remote user shall be required to log into the system via the web client. The login is for accessing the Test Server. [1]
- 2.7.2 The system shall authenticate the user. If the user is authenticated, they are allowed to proceed. If authentication cannot be made, an error message describing the issue is displayed. [1]
- 2.7.3 The user shall have the ability to register their local test engine with the test server. [1]
- 2.7.4 The user shall have the ability to de-register their local test engine with the test server. [1]

Requirements (User and System Level)

- 2.8 The system shall be highly available and disaster recoverable. The system will have a production environment that is usable by multiple users implemented in multiple regions and availability zones in the cloud. [2]

- 2.8.1 The user shall have the ability to install the test engine on multiple machines (redundancy, performance, latency). [0]
- 2.8.2 The system shall maintain all program code in scripts that can be deployed to the cloud platform. 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. [2]
- 2.8.3 Backup copies of all scripts shall be located in a separate region. [2]
- 2.8.4 System source code shall be deployed to cloud instances hosted in several regions and availability zones. [2]
- 2.8.5 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. [1]
- 2.8.6 The test environment shall be implemented in multiple zones and multiple regions to enable testing of HA/DR requirements rather than taking production down. [3]

Availability and Business Continuity

- **3.1 Availability Requirement 1: Continuous System Uptime [1]**
- 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.
- **3.2 Availability Requirement 2: Recovery Time [1]**
- 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.
- **3.3 Availability Requirement 3: High Availability [1]**
- The system shall support high availability by being quickly accessible to users attempting to access it from any geographic region.

Technical Constraints

- 4.1.1 The system shall be developed using the C++ programming language and the C++ Standard Template Library (STL). [0]
- 4.1.2 The system shall be developed using a publicly available source code editor which supports the C++ language. [0]
- 4.1.3 The system shall have at least 75% unit test coverage of the source code. [1]

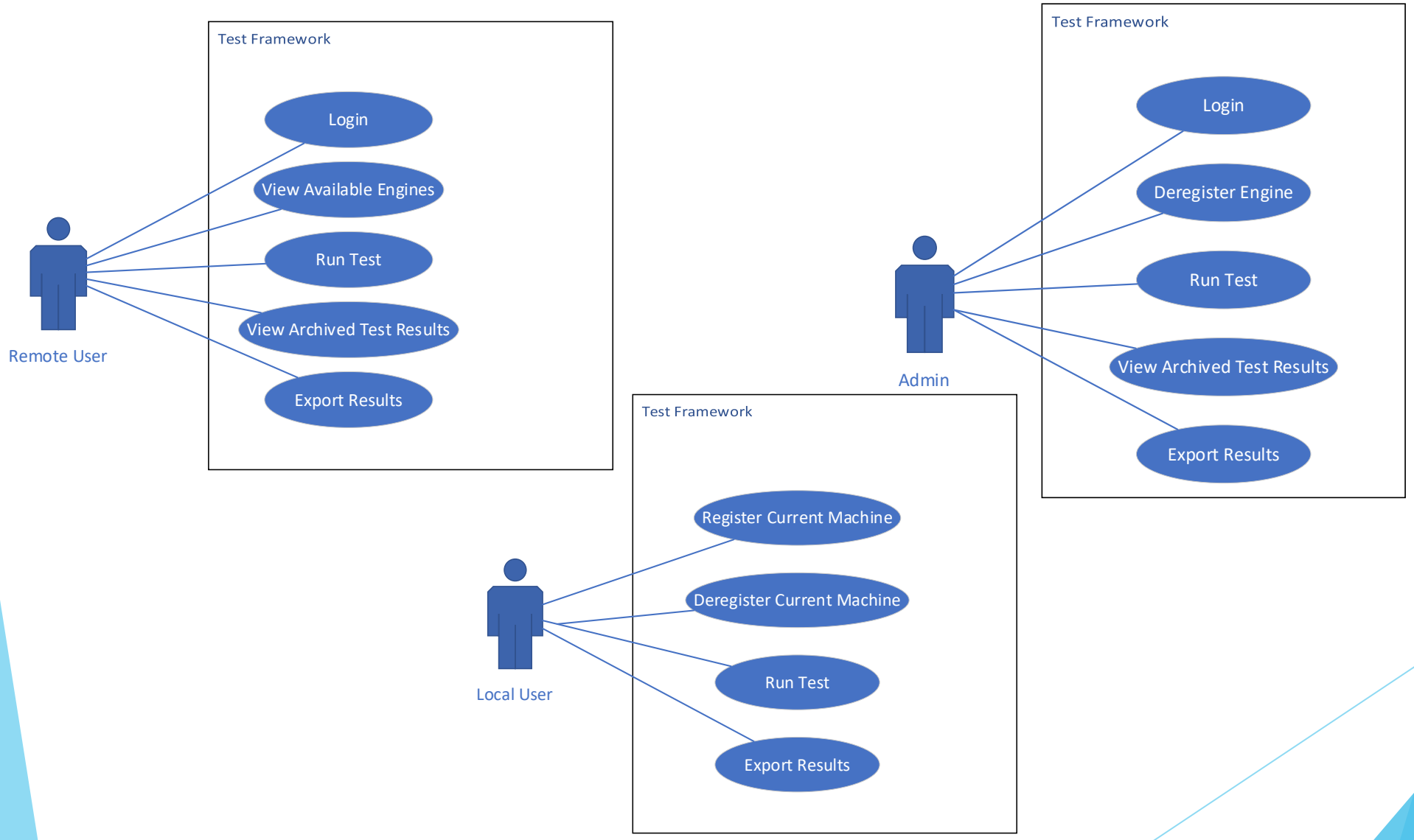
Operational Constraints

- 4.2.1 Granting access to a new user of the system shall take no more than 1 business day to complete. [0]
- 4.2.2 Modifying or removing a user's access to the system shall take no more than 1 business day to complete. [0]

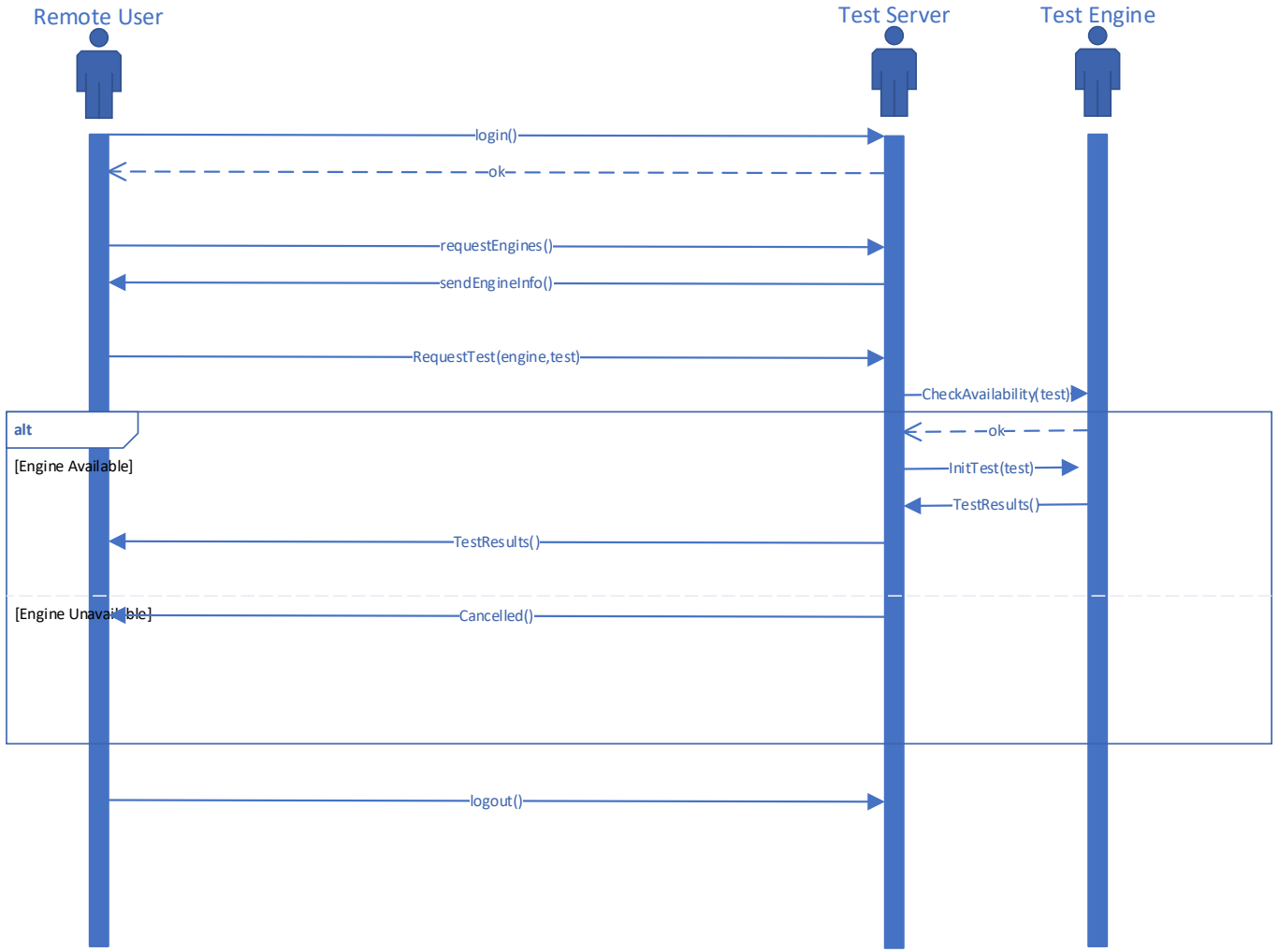
Business Constraints

- 4.3.1 Disaster recovery shall be cost-effective and managed through the fault tolerance and high availability features of the cloud-based system architecture. [1]
- 4.3.2 User training shall take no more than 1 business day to complete, regardless of the user's role. [2]

System Models: Use Case

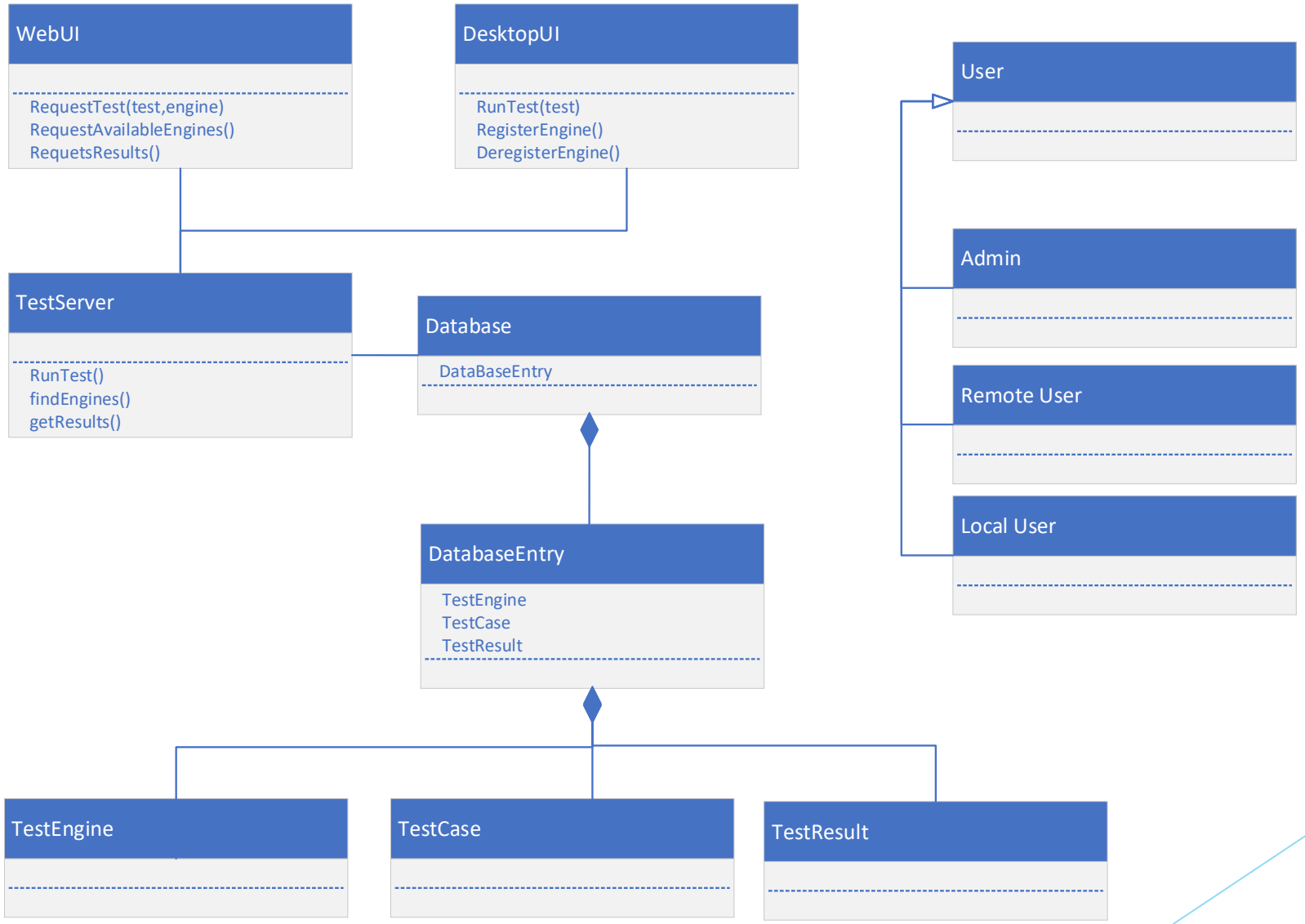


System Models: Sequence Diagram



Run Remote Test

System Models: Class Diagram



Thank You

- Questions?

Backup

Availability and Business Continuity

- **3.1 Availability Requirement 1: Continuous System Uptime**
- 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. [1]
- 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. [1]
- **3.2 Availability Requirement 2: Recovery Time**
- 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. [1]
- 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. [1]
- 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. [1]
- 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. [1]

Availability and Business Continuity

- **3.3 Availability Requirement 3: High Availability**
- The system shall support high availability by being quickly accessible to users attempting to access it from any geographic region. [1]
- 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. [1]
- 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. [1]
- 3.3.3 The system shall maintain all program code in scripts that can be deployed to the cloud platform. [1]
- 3.3.3.1 Backup copies of all scripts shall be located in a separate region. [2]