Asking (Question and Answer Platform)

Master Test Plan

Version 1.0

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 05/05/2017 | 1.0 | Master test plan | P.R.L. Fernando |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Evaluation Mission and Test Motivation 3

1.1 Background 3

1.2 Evaluation Mission 3

1.3 Testing Motivators 3

2. Target Test Items 4

3. Test Approach 6

3.1 Testing Techniques and Types 6

3.1.1 Data and Database Integrity Testing 6

3.1.2 Function Testing 8

3.1.3 User Interface Testing 9

3.1.4 Performance Profiling 10

3.1.5 Load Testing 11

3.1.6 Security and Access Control Testing 12

3.1.7 Failover and Recovery Testing 13

3.1.8 Configuration Testing 15

4. Deliverables 16

4.1 Test Evaluation Summaries 17

4.2 Reporting on Test Coverage 17

5. Risks, Dependencies, Assumptions, and Constraints 17

6. References 18

Master Test Plan

# Evaluation Mission and Test Motivation

This test plan was developed as a guide to the asking (question and answer platform) testing effort. This test plan is to gather all of the information necessary to plan and control test effort for testing asking platform. If describes the approach to testing the software, and tools and techniques that will be used during the testing process. Moreover, the test plan will provide a methodology on what should test and the types of tests that have to perform. Finally, the test plan will enable guide us to release a stable and bug-free version of the software.

## Background

The third phase of the project involves implementing asking question and answer platform based on the requirement and design documents of the inception and elaboration phases. The asking platform will be developed by the implementation using java script. Node.js with Express.js is used as the backend framework and angular2 used as the frontend framework while mongodb is used as the database engine. A comprehensive test plan need to ensure that the platform conforms to the specification, design and to perform quality assurance on the final product. This will enable the release of a complete and bug free software and minimize the risk of software failures.

## Evaluation Mission

The main objectives of the construction phase are to ensure that the specification of the requirement document have been achieved, to ensure that the specifications of the design document have been achieved and to ensure that the risk of software failure is reduced to a minimum.

This document is to achieve these objectives and to verify that these objectives have been met. Meeting these objectives will enable to release a stable version of software.

## Testing Motivators

The targeted test items listed below will be the motivation for testing in this phase.

**Data and Database Integrity Testing**: Verify that data is being stored by the system in a manner where the data is not compromised by updating, restoration or retrieval processing.

**Function Testing**: Will ensure that the use cases have been met.

**User Interface Testing**: Will verify if the requirements of the GUI have been implemented as specified.

**Performance Profiling**: Ensure that the website performance is at an acceptable level when searching and modifying data.

**Load Testing**: See how the website performs when the total number of active users at its limits.

**Security and Access Control Testing**: Will endure that unauthorized access is prevented.

**Failover and Recovery Testing**: Will test and simulate whether the system will recover from failures.

**Configuration** **Testing**: Ensure that the website functioning correctly under different web browsers.

# Target Test Items

The listing below identifies those test items: software, hardware, and supporting product elements that have been identified as targets for testing. This list represents what items will be tested.

**Function Testing**

Function testing consists of testing all the requirements and specifications, as per the requirements and specifications document. In essence, the list of functions to test corresponds to the list of use cases and requirements in the requirements document

1. Create lecturer accounts
2. Remove lecturer accounts
3. Blacklist student accounts
4. Add new module
5. Modify modules
6. Login to the system
7. Register new user
8. Subscribe for a module
9. Search questions
10. View questions and answers
11. Submit new question
12. Submit new answer
13. Rate a question
14. Rate an answer
15. Modify an answer
16. Delete a question
17. Delete an answer
18. View a statistical report of questions and answers submitted

**User Interface Testing**

User interface testing is concerned with making sure that each functionality concerning the user interface is works as per the requirements defined in the design document. For the user interface, the possible interactions with the website will be tested in great detail. During the test, the objective will be to compare and check the validity of an implemented functionality with the expected functionality. Below is a list of the User Interface items that will be tested:

1. Home UI
2. Admin Panel UI
3. Search Question UI
4. Add New Question UI
5. Add New Answer UI
6. View Question UI
7. Modify Question, Answer UI
8. Browse Modules UI
9. View Module Details UI
10. Add, Manage Users UI
11. Login UI
12. Sign Up UI
13. Update Account UI
14. View Report UI

**Performance Profiling**

Performance profiling is concerned with testing the different response times of the software. In these types of tests, focused mainly on the following test items:

1. Search result response time
2. Add new question or answer response time
3. Generate new report response time

**Load Testing**

Load Testing is concerned with testing the system beyond the limits it was designed for. In this type of test, focused mainly on testing the website when high number of active users are accessing the system. Below are the test items that were identified

1. Functionality of website with high number of active users are accessing the system
2. Search result response time
3. Create new report response time

**Configuration Testing**

Configuration testing is concerned with testing the system under different environment configurations. In this type of test, focused on testing the website under different platforms (mobile, desktop) different web browsers and different versions. Below is a list of platforms and web browsers the website will be tested under

**Platforms**

1. Mobile
2. Desktop
3. Tablet

**Web** **browsers**

1. Google chrome
2. Mozilla Firefox
3. Microsoft Edge
4. Microsoft Explorer

# Test Approach

The Test Approach describes the recommended strategy for designing and implementing the required tests. In this section, details of the tests that need to be performed for each target test item that was identified will be described. These tests will be organized into the following sub-sections

1. Data and database integrity testing
2. Function Testing
3. User Interface Testing
4. Performance Profiling
5. Load Testing
6. Security and Access Control Testing
7. Failover and Recovery Testing
8. Configuration Testing

For each of these test motivators, test objective test approach and strategies are described in details.

## Testing Techniques and Types

### Data and Database Integrity Testing

Data and database testing will test individual methods which attract with database with their functionalities in isolation. This low-level testing will test database access methods and process independent of the user interface.

|  |  |
| --- | --- |
| Technique Objective: | The objective of data and database integrity testing is to test database access methods and process independent of the user interface and to observe and log incorrect functioning target behavior or data corruption and ensure the soundness, completeness and wholeness of data stored. |
| Technique: | In data and database integrity testing each database access method and process will be invoked each seeding with valid and invalid data or requests for data. Moreover, database will be inspected to ensure the data has been populated as intended and all database events have occurred properly and reviewing the returned data to ensure that the correct data was retrieved for the correct reasons. |
| Oracles: | The strategies used to identify and accurately observe the outcomes of test are as follows.   1. Verify that, when a particular set of data is saved to the database, each value gets saved fully, and the truncation of strings and rounding of numeric values do not occur. 2. Checking whether the database collection schemas can be modified and deleted. 3. Running data tests for all data types in all database tables and reviewing the returned data. 4. Check whether blank values can be retrieved from the database 5. Verify that the default values are saved in the database, if the user input is not specified. 6. Verify whether indexes are updated correctly. 7. Verify whether text indexes working correctly after inserting data. |
| Required Tools: | 1. Mocha.js for test generation and test automation 2. Chai.js assertion library to compare results and generate error reports. 3. Robomongo (mongodb client) to view collections and documents. 4. MongoDump and MongoRestore for database backup and restore. 5. Faker as the fake data generation tool to use with the testing process. |
| Success Criteria: | All the invoked database process run successfully and outcome the desired data output. |
| Special Considerations: | 1. Tester should have a good knowledge on NoSQL databases. And mongodb query language. 2. Testing may require DBMS development environment or drivers to enter or modify data directly in the database. 3. Small or minimally sized databases should be used to increase the visibility of any non-acceptable events. 4. Process should be invoked manually. |

### Function Testing

This section is concerned with testing the functions (or requirements) of the software. This is a critical aspect of the testing effort, as it ensures that the software meets the requirements, and thus ensures acceptance by the users. For completeness, each requirement should be associated with a set of test cases, some with valid data, and some with invalid data. The goals of these tests are to verify proper data acceptance, processing and retrieval, and appropriate implementation of the business rules.

|  |  |
| --- | --- |
| Technique Objective: | The objective of the function testing is to ensure that the software meets the requirements, and thus ensures acceptance by the users and to verify data acceptance, processing and retravel, and appropriate implementation of business rules. |
| Technique: | 1. In functional testing, each use case is to be tested using both valid and invalid data to verify the occurrence of expected results and success messages to valid data and occurrence of appropriate error messages to invalid data. 2. Black box tests and consider only about inputs and outputs of functions. 3. Validation of the input data is to be tested |
| Oracles: | The strategies that can be used to identify and accurately observe the outcome of the test are as follows.   1. The identification of functions that the system is expected to perform. 2. The creation of input data based on the function's specifications 3. The determination of output based on the function's specifications 4. The execution of the test case 5. The reviewing of actual and expected outputs 6. To check whether the application works as per the end-user need |
| Required Tools: | 1. Protractor as the test automation tool for end to end testing. 2. Karma as the test runner for unit testing and function testing of angular components and angular services. |
| Success Criteria: | All the key use case scenarios and features relevant to their execution process and output desired results and validation of input data that happen inside the user interface working correctly. |
| Special Considerations: | For testing the add new question, only the modules names and topics available in the database can be used since the function has been implemented to get the module names and topics from a pick list.  Adding, modifying and removing data will invoke a confirmation messages. |

### User Interface Testing

To test the User Interface, each functionality described in the design document will be verified to see if it has been implemented correctly, if it responds normally and also if no errors occur during the process between the user and the platform. A schema will be used to test (unit), what is the purpose of the test (what is tested), what are the inputs (from the user for instance), what is the expected result and also what is the effective (real result).

The user interfaces available for the testing of the Asking question and answer as follows,

1. Home UI
2. Admin Panel UI
3. Search Question UI
4. Add New Question UI
5. Add New Answer UI
6. View Question UI
7. Modify Question, Answer UI
8. Browse Modules UI
9. View Module Details UI
10. Add, Manage Users UI
11. Login UI
12. Sign Up UI
13. Update Account UI
14. View Report UI

|  |  |
| --- | --- |
| Technique Objective: | Objective of UI testing is to observe and log standards conformance and target behavior of the application. The verification of navigation through the application reflecting business functions and requirements, including window-to-window, field-to- field, and use of access methods (tab keys, mouse movements, accelerator keys will be done in UI testing. I addition the characteristics of the window objects such as menus, size, position, state, and focus also can be tested in UI testing |
| Technique: | For each and every window, create test cases to test   1. Check and verify screen elements 2. Navigation between web pages 3. Inserting and selecting data 4. Submitting form requests |
| Oracles: | The strategies that can be used to identify and accurately observe the outcome of the test are as follows.   1. The identification of functions that the system is expected to perform. 2. The creation of input data based on the function's specifications 3. The determination of output based on the function's specifications 4. The execution of the test case 5. The reviewing of actual and expected outputs   To check whether the application works as per the end-user need |
| Required Tools: | 1. Firefox browser with Selenium IDE as a test automation tool 2. Protractor to test end to end functions and to automate testing. 3. Firefox XPATH plugin to easily identify and select elements of the web site. |
| Success Criteria: | All the user interface navigations working correctly and all the check list items in the test cases are marked as verified. |
| Special Considerations: | Testing tool must have configured to login to the system as different kind of access levels (different kind of users) because each user has a set of unique pages. |

### Performance Profiling

Performance profiling is done to measure and evaluate the performance of the application including response time, data access rate and other time-sensitive requirements

|  |  |
| --- | --- |
| Technique Objective: | Objective of performance profiling is to exercise behaviors for designated functions to observe and log target behavior and application performance data. In order to undertake a successful testing, application should be tested under both normal anticipated workload and anticipated worst-case workload. |
| Technique: | The technique used to achieve the performance profiling is an enhanced version of the use test procedures. The test cases developed for function or business cycle testing are used with modifying data files to increase the number of transactions or the scripts to increase the number of iterations that occur in each transaction. The scripts should be run on one machine and should be repeated with multiple clients. |
| Oracles: | The strategies that can be used by the technique to accurately observe the outcomes of the test are as follows.   1. Testing under normal anticipated workload and anticipated worst-case workload. 2. Determine output based on the workload. |
| Required Tools: | Test techniques as follows   1. Selenium IDE as the test automation tool 2. Node.js inbuild profiling tool to performance profiling |
| Success Criteria: | The success criteria vary depending on the no. of users and no. of transactions. The success criteria for single transaction or single user are the successful emulation of the transaction scripts and for the multiple transactions or multiple users, successful emulation of the workload without any failures. |
| Special Considerations: | Performance testing includes having a background workload on the server. Methods to achieve work load on the server is as follows.   1. “Drive transactions” directly to the server, usually in the form of Structured Query Language (SQL) calls. 2. Create “virtual” user load to simulate many clients, usually several hundred. Remote Terminal Emulation tools are used to accomplish this load. This technique can also be used to load the network with “traffic”.   3. Use multiple physical clients, each running test scripts, to place a load on the system. |

### Load Testing

Load testing is a performance test that subjects the target-of-test to varying workloads to measure and evaluate the performance behaviors and abilities of the target-of-test to continue to function properly under these different workloads. The goal of load testing is to determine and ensure that the system functions properly beyond the expected maximum workload. Additionally, load testing evaluates the performance characteristics, such as response times and other time-sensitive issues. Most of the times the bottleneck is memory.

|  |  |
| --- | --- |
| Technique Objective: | Objective of load testing is to exercise designated transactions or business cases under varying workload conditions to observe and log target behavior and system performance data. |
| Technique: | 1. Use Postman client to make requests to web pages and test how fast the system loads the views to the requested route. 2. Use selenium IDE to trigger a large number of requests to test whether the system still responsive to a new user. The numbers should simulate a real world worst case scenario. 3. Write test cases to run queries multiple times using a loop to evaluate the performance of the database. 4. Workloads should include both average and peak loads 5. The workloads should be executed under different test environments |
| Oracles: | 1. All the measures will be simulated in the program code itself since it is it is difficult to exhaust a modern computer system manually. 2. Still the system performance may vary when the system is hosted in a remote cloud due to the performance of the cloud computers and the strength of the connectivity of the user |
| Required Tools: | 1. Mocha.js as test automation tool 2. Chai.js as the assertion library 3. Mongodb reporting tools to evaluate the database performance. 4. Postman client to send requests to the server 5. Protractor as the end to end test automation tool |
| Success Criteria: | 1. The success criteria for single user are the successful emulation of the multiple users, successful emulation of the workload without any failures. 2. Exhausting the system will result in the identification |
| Special Considerations: | 1. Load testing should be performed on a dedicated machine or at a dedicated time. This permits full control and accurate measurement. 2. The databases used for load testing should be either actual size or scaled equally. |

### Security and Access Control Testing

Security and Access Control Testing focuses on two key areas of security:

1. Application-level security, including access to the Data or Business Functions
2. System-level Security, including logging into or remotely accessing to the system.

Application-level security ensures that actors are restricted to specific functions or use cases, or they are limited in the data that is available to them.

System-level security ensures that only those users granted access to the system are capable of accessing the applications and only through the appropriate gateways.

|  |  |
| --- | --- |
| Technique Objective: | Objective of security and Access control testing is to exercising the test on application level security and system level security.   1. Application-level Security: An actor can access only those functions or data for which their user type is provided permissions. Objective is to ensure that only permitted function or data can access by the user. 2. System-level Security: Only those actors with access to the system and applications are permitted to access them. Objective is to ensure that users who has not access to the system are not permitted to access them. |
| Technique: | **Application-level Security**: Identify and list each user type and the functions or data each type has permissions for.   1. Create user instances with different roles and try to perform different tasks 2. Ensure authorized users are given permission to perform the required tasks and unauthorized tasks are detected. 3. Modify the role of the users created and try out performing the tasks and ensure the additional functions are correctly available or denied.   **System-level Access**:   1. Write scripts to perform simple tasks like creating new table dropping a table, altering database and make sure they are not permitted. 2. Write simple tasks to get security data from data by writing queries that the website not support and make sure they are nor executed. |
| Oracles: | The strategies that can be used to identify and accurately observe the outcomes of the test are as follows   1. The identification of accesses to system functionalities based on the role of the user. 2. The creation of input data based on the users’ access permission. 3. The determination of success in access based on the user type. 4. The comparison of actual and expected outputs.   The testing strategy consists of both automation testing and manual testing. Still there may be some security loopholes making the system vulnerable to attackers. XSS attacks can still occur and these security loopholes need to eliminate. |
| Required Tools: | The technique requires the following tools   1. Selenium IDE can be used in order to use as a test automation. 2. Postman client can be used to manually make http requests to the serer and to verify outputs. 3. Source clear can be used to search for vulnerable packages and to update them. 4. Node security platform can be used to check for the vulnerabilities in the source code. |
| Success Criteria: | The success criteria can be taken upon the access to appropriate functions based on the user role. |
| Special Considerations: | Access to the system must be reviewed or discussed with the appropriate network or systems administrator. |

### Failover and Recovery Testing

Failover and recovery testing ensures that the target-of-test can successfully failover and recover from a variety of hardware, software or network malfunctions with undue loss of data or data integrity. For those systems that must be kept running failover testing ensures that, when a failover condition occurs, the alternate or backup systems properly “take over” for the failed system without any loss of data or transactions. Recovery testing is an antagonistic test process in which the application or system is exposed to extreme conditions, or simulated conditions, to cause a failure, such as device Input/output (I/O) failures, or invalid database pointers and keys. Recovery processes are invoked, and the application or system is monitored and inspected to verify proper application, or system, and data recovery has been achieved.

|  |  |
| --- | --- |
| Technique Objective: | Simulate the ability of the system to come back to a desired known state and database to restore after a condition where system goes down. The following types of conditions are included in the testing to observe and log target behavior after recovery,   1. power interruption to the server 2. communication interruption due to network failures 3. incomplete cycles (data filter processes interrupted, data synchronization processes interrupted) 4. database failures due to data overloads or large number of concurrent connections 5. corruption of data in the database due to damages in the physical disk and false queries 6. Crashes occurs on the web server. 7. Crashes occurs on the database server. |
| Technique: | The tests already created for Functional testing can be used for the testing under this category since the testing is done to ensure the system functions in order after the recovery.   1. Power interruption to the server: simulate or initiate power down procedures for the server. 2. Interruption via network servers: simulate or initiate communication loss with the network (physically disconnect communication wires or power down network servers or routers). 3. Disconnect the database connectivity by shutting down the database server and operate the system to check whether a valid error message prompts to the end user. 4. Use Nginx web server and simulate or initiate server crashes. 5. Use a replica set and test and simulate mongodb database failures and crashes.   Once the above conditions or simulated conditions are achieved, additional transactions should be executed and, upon reaching this second test point state, recovery procedures should be invoked. Testing for incomplete cycles uses the same technique as described above except that the database processes themselves should be aborted or prematurely terminated.  Testing for the following conditions requires that a known database state be achieved.  Several database fields, pointers, and keys should be corrupted manually and directly within the database (via database tools). Additional transactions should be executed using the tests from Application Function and Business Cycle Testing and full cycles executed. |
| Oracles: | To simulate the failure environment, it requires some tasks to be performed manually. For example, by making use of a user to switch off the system and unplugging network interfaces in order to simulate the loss of power or network.  Still there are particular failure environment whose simulation costs more than the implementation of the system.  It is essential to have a backup mechanism where data is backed up to an isolated premise form where data can be extracted in a total system failure. |
| Required Tools: | The technique requires the following tools:   1. Installation monitoring tools and system monitors like system monitor and task manager to measure the resource utilization and memory dumps performed before and after the failure. 2. Use of backup tools like MongoDump and Robomongo to back up and restore the system schema and data. |
| Success Criteria: | The technique supports the testing of   1. One or more simulated disasters involving one or more combinations of the application, database, and system. 2. One or more simulated recoveries involving one or more combinations of the application, database, and system to a known desired state. 3. This will motivate the system to build in a robust operating environment since the tests show the possibility of failures to occur. |
| Special Considerations: | 1. Recovery testing is highly intrusive. Procedures to disconnect cabling (simulating power or communication loss) may not be desirable or feasible. Sometimes it will cost more than the system implementation. Alternative methods, such as diagnostic software tools may be required. 2. Likelihood or probability of such disaster. 3. Cost due to outages and cost of protecting the system (target-test). 4. Resources from the Systems (or Computer Operations), Database, and Networking groups are required. 5. These tests should be run after hours or on an isolated machine. |

### Configuration Testing

Configuration testing verifies the operation of the target-of-test on different software and hardware configurations. In most production environments, the particular hardware specifications for the client workstations, network connections, and database servers vary. Client workstations may have different software loaded for example, applications, drivers, and so on and, at any one time, many different combinations may be active using different resources.

|  |  |
| --- | --- |
| Technique Objective: | The objective of configuration testing is to exercise the target-of-test on the required hardware and software configurations to observe and log target behavior under different configurations and identify changes in configuration state. |
| Technique: | 1. Use Function Test scripts to ensure the database connection and verify the availability of permissions required by the system. 2. Open and close various non-target-of-test related software, either as part of the test or prior to the start of the test. 3. Execute selected transactions to simulate actors interacting with the target-of-test and the non-target-of-test software. 4. Repeat the above process, minimizing the available conventional memory on the client workstation. |
| Oracles: | 1. Access web site with different web browsers and verify that the web page is displayed correctly and functioning correctly. 2. Since angular.js used as the front-end framework it need to verify its functionalities with different web browsers. 3. For a web server, the requirement for configuration testing is minimal since it is guaranteed the application would run in a remote server with Google chrome v8 engine. |
| Required Tools: | 1. Node.js 6.5+ enabled 2. Mongodb 3.4.4+ enabled |
| Success Criteria: | This will enable to finalize the required environment to deploy the system so that the users of the system can successfully access the system. |
| Special Considerations: | 1. End users are not required to configure anything related to the web server. 2. End users need configure their web browsers and their internet connections properly. |

# Deliverables

The list of artifacts that will be created by the test effort that are useful deliverables to the various stakeholders of the test effort are as follows.

**Test plan**

This is the master test plan which cover all the use case of the system and their functionalities.

**Test case documents**

By reading test case document stakeholders get an idea about the quality of test cases written and the effectiveness of those test cases. Stakeholders can also provide inputs about the current set of test cases as well as suggest some more missing test cases.

**Testing strategy**

How testing would be carried out and defines test approached.

**Test scripts**

Methods and function written using testing languages and frameworks to verify that the system performs as expected.

**Test inputs/data**

Inputs and data used to test the system.

**Test outputs/results/reports**

Results and reports obtained by running test cases. These results describe the environmental or operating conditions, and shows the comparison of test results with test.

## Test Evaluation Summaries

The defined test cases for asking question and answer platform will be executed following the test strategy.

Test coverage refers the covering of function related to use cases and test requirements.

Following are the summaries and reports that need to be created regarding the testing.

1. Karma testing summary to check angular components and front end logic.
2. Protractor testing summary to check end to end testing
3. Mocha test summaries to check node.js functionalities.
4. Browser testing summaries
5. System monitoring and memory monitoring summaries
6. Code inspection summaries
7. Selenium IDE user interface testing summaries.
8. Performance profiling summaries for node server and database.
9. Database statistics
10. Security breach analysis summaries

## Reporting on Test Coverage

The testing will happen at the end of every phase and the testing results will be delivered. Functionality testing will happen in parallel with the completion of each of the functional requirements. The test status will be included in a table as follows.

|  |  |  |
| --- | --- | --- |
| Function | Status | Coverage |
| Search question | Completed | 100% |
| Add new question | Test | 50% |
| Add new answer | TO be tested | 0% |

Load testing and security testing will be carried out once the implementation is completed.

# Risks, Dependencies, Assumptions, and Constraints

| **Risk** | **Mitigation Strategy** | **Contingency (Risk is realized)** |
| --- | --- | --- |
| Prerequisite entry criteria are not met. | Tester will define the prerequisites that must be met before Load Testing can start.  End user will endeavour to meet prerequisites indicated by Tester. | 1. Meet outstanding prerequisites 2. Consider Load Test Failure |
| Test data proves to be inadequate. | End user will ensure a full set of suitable and protected test data is available.  Tester will indicate what is required and will verify the suitability of test data. | 1. Redefine test data 2. Review Test Plan and modify 3. components (that is, scripts) 4. Consider Load Test Failure |
| Database requires refresh. | System admin will endeavour to ensure the Database is regularly refreshed as required by Tester. | 1. Restore data and restart 2. Clear Database |
| Web site becomes unavailable | Testing will be delayed until this situation is rectified | 1. Connect with web service provider 2. Establish on branded web service provider |
| Web testing software is not available/does not work | This will delay the introduction of automated testing and result in more manual | 1. Reinstall the testing tool 2. Configure testing tool correctly |
| Not enough time to complete all test cases | If time cannot be extended, individual test cases will be skipped, starting with the lowest priority | 1. Consider most valuable functions that should tested such as admin data input and passenger search bus schedule |

# References

* Indicate the tool references (name, available at <<URL>>) used for testing
* Refer any data/ information in a standard format (eg. IEEE referencing style)
* For different algorithms/ techniques/ theories you can refer text books.
* For tools you can refer web pages.
* For similar work you can refer research paper articles that describe the work.
* You may include white paper articles for the description of technologies; web URL for the tool references. When you refer such a web page, you have to indicate the (Accessed on <<date>>)