WHITE PAPER

ENHANCING MANUAL TESTING

ABSTRACT

Manual testing is an important part in the software testing lifecycle of any product. However, effective manual testing cannot be obtained by functional verification alone. It needs to be enhanced with the help of advanced testing types, by leveraging different testing techniques and by employing effective test management.

This paper is part one on this series of increasing the effectiveness of manual testing and will focus on using advanced testing types. This advanced validation helps the tester to increase the robustness of the product under test and improve the test coverage. This enhances test efficiency and helps achieve overall testing quality to deliver a robust product and optimize manual testing efforts.

TABLE OF CONTENTS

Introduction 3
Manual Testing Technique 3
Types of Testing 3
Advanced Validation Techniques 4
Implementing Advanced Validation 4
- Integration Testing 4
  - Middle Tier Verification 6
  - Database Verification 6
  - Logs Verification 6
- Performance Testing 6
  - Monitoring Page Load Time 6
  - Monitoring response Time 7
  - View Source 7
  - Checking File Size Optimization 7
- Security Testing 8
  - Session Management 8
  - Secure Transmission and Authentication Tests 8
  - URL Manipulation Testing 8
  - Error Handling 8
- Compatibility Testing 8
  - Browser Compatibility 8
  - Hardware Compatibility 9
  - Locale Compatibility 9

Conclusion 9
References 9
About the Author 10

INTRODUCTION

Increase in competition and leaps in technology have forced companies to qualitatively assess their products before launching it in the market. This has put a lot of impetus on software testing and an ever increasing need to focus on rigorous testing of software products. With the arrival of many automation tools in market, some may argue towards the need of Manual effort.
Manual Testing, however, remains a strong pillar in the assessment and verification of any software product.

Having said that, testers do need to go beyond the regular functional aspects of manual testing. We can achieve much more by smartly adding advanced testing types, leveraging different testing techniques and employing effective test management.

This paper is part one in this series and will focus on what are advanced testing types and how they can be put to use in existing manual testing structure to reap extra benefits.

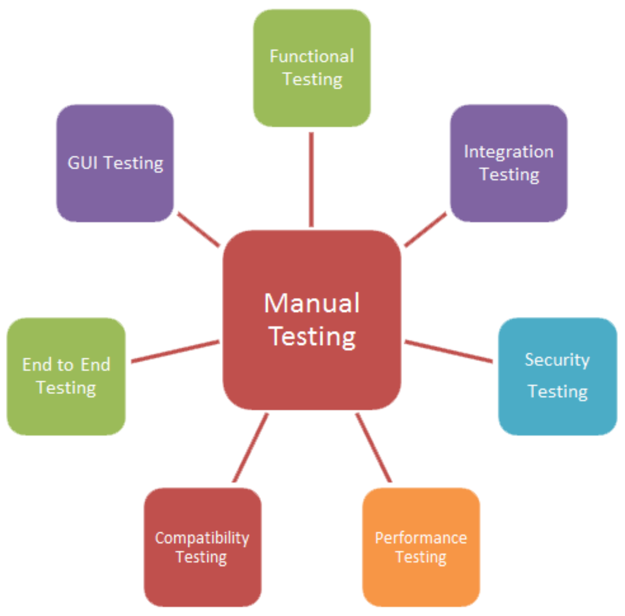
MANUAL TESTING TECHNIQUE

As per Wikipedia, Manual Testing is the process of manually testing software for defects. It requires a tester to play the role of an end user and use most of all features of the application to ensure correct behavior.
This, however, is a very narrow definition and majorly focuses on functional aspects of manual testing.

In this paper we will see how manual testing encompasses much more than just functional verification of the product under test. We will examine how this effort can be extrapolated over multiple areas to deliver a high-quality, robust product and improve the overall test-coverage.

We will study how different types of advanced testing can be coupled along with functional and UI validations to achieve greater success.

TYPES OF TESTING



As the above image showcases, Manual Testing has a vast paradigm ranging from functional testing to GUI testing and so on. However, in reality, majority of the software testers only focus on below types when performing manual testing:

- Functional verification on new features.
- Regression on existing features.
- Verification of end-to-end business flows from end-user perspective.
- GUI testing.
- User Acceptance testing.

The above provides a good quality product but it can be made better by adding the advanced validation techniques namely:

- Integration Testing
- Security Testing
- Performance Testing
- Compatibility Testing

These have been explained in detail in further sections below.

ADVANCED VALIDATION TECHNIQUES

To enhance the efficiency of manual testing, we can leverage above mentioned advanced validation techniques.
Although these exist as independent testing techniques and give very good results when used in conjunction with automated test tools, we can involve these at granular level to improve the overall scope of manual testing.

In next topic, we will gain insight into how exactly and to what extent these can be implemented as part of manual verification effort.

IMPLEMENTING ADVANCED VALIDATION

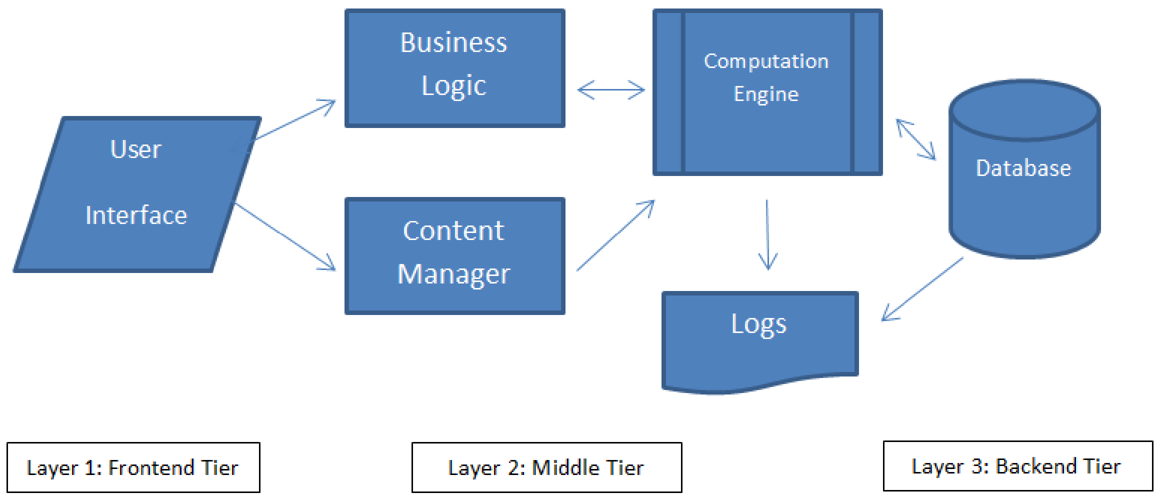
Let us understand these techniques one by one and how we can use them in our day to day manual testing effort.
Majority of these are being used in my current project and the results are much superior to functional testing alone.

INTEGRATION TESTING

A lot of QA engineers believe that Integration testing is a white box testing technique involving unit testing and writing small code snippets to test blocks of the application under test. This is true but only to a certain extent.
Nowadays, all major applications have certain building blocks or modules which communicate with each other to form a flow. These components can be tested individually using certain black box techniques as part of Integration testing.

Let's take an example here:

There is a dining web portal where users enter their personal details like age, location, cuisine preference. These details are parsed via the middle tier business layer and appropriate matching dining options are presented to the user.
The typical high-level architecture would be something like:



Quick definitions of the various modules below:

Frontend layer

The frontend layer is mainly the modules exposed to the end user. This comprises of the User Interface module. It is typically the UI page/portal where all the inputs are given regarding user's age, location, cuisine preference.

Middle Tier layer

The Middle tier layer is usually the engines and components which connects the frontend to the backend and where all computations and logics are used. It consists of below modules:
Business Logic module which has all the business specific methods defined based on which certain calculations are made and a query is fired to derive proper results.
Content Manager module which has all the static contents, specific to the inputs entered by the user.
Computation Engine module is where all the calculations are made based on the data that comes from business and content modules. Based on these calculations, a SQL query is triggered into the database.

Backend layer

The backend layer is the database containing all user records, list of dining options and other related data.
In addition is the Logs module which refers to the logging management for all events, error, warnings and information.

Now, based on the architecture of this project, we can segregate our testing module-wise and ascertain specific techniques which are relevant to each module:
Since, the frontend here is a UI module, majorly the testing would comprise of UI testing and Functional testing. This is a pretty self-explanatory form of manual testing so we will not go into details here.
Let's examine the other layers and their verification henceforth.

MIDDLE TIER VERIFICATION

The middle layer in our example is the major building block channeling the flow of data from UI to backend and vice-versa. So, from integration testing perspective, testing whether the information / data is flowing correctly is essential.
Let's say that the communication of data is done in XML files. Main testing pointers would be to check if the files are generated with correct data? Is the data being transferred from one module to another? All these things will be tested as part of Integration testing.
The tester can try to update the XML tags and check the behavior. Also, if certain webAPIs are being called, the tester can use REST client or SOAPUI to analyze the response of the various requests being made.

DATABASE VERIFICATION

Once the XML format has been tested, next step is to make sure correct data is being recorded at the database level.

Simple SQL queries can help the tester verify the authenticity of the data being added at the backend.
Also, when performing any search or retrieval of existing data on UI, tester should run the same query at DB level and cross-check against the UI results.

It can be taken a step further, by doing some database level insertions and verifying the result on GUI. Some applications sync the data and UI reflects the DB level changes. In others, there should a graceful handling of such data injection.

LOGS VERIFICATION

Logging is an important part of product lifecycle. Logs usually contain any logged events, such as verification point failures, script exceptions, object recognition warnings, and any additional error information. In some cases, information messages are also part of diagnostic log.
Most of the testers ignore logs as they think that checking of logs falls under the territory of a developer for debugging purposes. However, if a tester starts analyzing logs at a basic level, it helps in a better understanding of any issues they have seen.

PERFORMANCE TESTING

Although most of the software projects have a dedicated performance testing team, who use advanced tools and scripts, some aspects of basic performance measurement (excluding load test) can be done by the manual tester.
We will take an example of a basic web portal where a user can search for health plans by providing their age, location, employment status and income.
Let us understand what all types of performance tests can be executed for this application.

MONITORING PAGE LOAD TIME

The basic performance test that can be performed throughout the application pages is to monitor the page load time. In the application under test, user is being asked to provide their location details. This might involve loading a map where user can point to their location. In general, pages with maps and image files, take slightly longer to load. Recording these can help the tester monitor the performance of the application and raise a red-flag if unusually higher load time is observed.
There are multiple browser plugins available which help easily recording this. In certain browsers, F12 or developer tools are present by default which is useful while analyzing the same.

MONITORING RESPONSE TIME

Along with page load time as explained in 5.2.1, another aspect is the response time. This is especially crucial if the search query is returning a lot of records.
For eg., in the application under test, when user provides the required details, they will be presented by a list of relevant health plans. In case, where number of matching records is high, response time can be higher than usual.
This might cause a performance issue. Ideally, pagination of results should be used so that performance is not impacted.

VIEW SOURCE

Since manual testers generally do not have access to code base, View Source comes handy when trying to check the source paths and domains of many links.
A special mention of viewing source when doing CDN testing. If the application uses CDN, we can check that all static content such as images, CSS, JS and videos is cached and accessed for the nearest CDN node.
This would essentially mean that domain for all above should be CDN and not the main test server.

CHECKING FILE SIZE OPTIMIZATION

In some web pages, images with bigger size are embedded, which cause delay while page load.
By using the F12 developer tools, tester can check the size of all the files found on the webpage.
These can then be compared against the standard file size and we can ensure whether they are optimized or can cause a performance lag in real time.

SECURITY TESTING

Security testing helps to identify the potential flaws in the system which can cause threat to data and functionality of the product. This testing exposes the vulnerability of the system and helps the coder to fix such gaps/leaks/issues.
There are many types of security testing and few of them can be strictly done using automation tools. However, we will focus on some aspects of security testing which can be performed manually while testing the application.

SESSION MANAGEMENT

Most of the web applications have end user login. In general, a user gets login credentials or registers on the website. These details are used for all future access of the website.

An important thing to test here is about session management of the user. These would typically include checking how the session management is handled in the application (e.g., tokens in cookies, token in URL). Tester can check the session tokens for cookie flags (httpOnly and secure).

Also, session termination after logout or session expiry should be verified.
Other tests can be to check if users can have multiple simultaneous sessions.

SECURE TRANSMISSION AND AUTHENTICATION TESTS

Verifying secure transmission means checking that login form, credentials and other such pages are delivered over HTTPS.

Other than that, an important part of manual verification is authentication testing. This includes testing all functionality for user passwords like password quality rules, remember me, CAPTCHA.

Also, user authentication functionality like testing for authentication bypass, user enumeration and cache management for user history is part of the Authentication testing scope.

URL MANIPULATION TESTING

In typical client-server model, the application uses the HTTP GET method to pass information. This information is passed in parameters in the query string appended usually in the base URL.
For testing purposes, the tester can modify a parameter value in the query string to check if the server accepts it. Ideally, we need to make sure that using URL manipulation, unauthorized user is not able to access the important information or not corrupting the database records.
For e.g., if the application has multiple user records and one user upon login can change their profile information, the tester needs to check that a particular user can only edit his/her information and by adding a different userid/username in URL they should not be able to access other users' s records.

ERROR HANDLING

Error handling involves checking for error codes and stack traces in the application. For an application under test, the error code should be descriptive enough to easily debug the issue.

Also, stack traces should be optimally used by tester to perform root-cause analysis.

COMPATIBILITY TESTING

With more and more companies catering to wide clientele, compatibility testing becomes a must. It ensures that the application under test runs across software's/hardware's/locales etc.
In addition, it helps identifying issues which are specific to a particular browser/language/system which are otherwise hard to find. This requires regression of the product against various different conditions.

BROWSER COMPATIBILITY

This types of testing essentially focuses on running and verifying the application on several or all supported browsers.
It may require some initial infrastructure setup as certain browsers can run only on certain operating systems.

The highlight of this testing is to not only test forward compatibility, but also backward compliance with earlier/older versions of browsers.

HARDWARE COMPATIBILITY

Similar to browser compatibility, hardware compatibility is about assessing that the application under test runs across all supported hardware configurations.

Again, some initial infrastructure is required for test setup. But once done, these are used for multiple test iterations.

LOCALE COMPATIBILITY

With more and more products becoming available globally, an important aspect comes as internationalization of the application to ensure its locale compatibility. More commonly known as localization testing, this is generally a bigger test effort and requires adequate test planning.
At a high level, locale specific content and features should be handled properly.
This is best taken up manually as it requires a lot of GUI testing because in different languages, the text-rendering and font formatting varies a lot.

In addition, many websites are tested for 508 compliance these days. This essentially means that users, regardless of disability status, can access the web portal. Although the Section 508 of the Rehabilitation Act has many compliance standards, majority of this testing includes using keyboard to perform certain functions on the website.

CONCLUSION

Most of the advanced testing types described here are employed in software projects but as individual testing units. These come very late in the testing cycle and usually after the functional testing is complete, the product is tested for performance, security and compatibility aspects.

However, if these validations are used in conjunction with the existing manual testing efforts, it can lead to early detection of major defects which will save the cost and efforts spent due to late detection.
It can also help in improving the overall scope of testing. This in turn will greatly enhance the test efficiency.

In conclusion, I strongly suggest that we should be implementing these testing types as part of manual testing (wherever the nature of the project allows it) to ensure a robust and high quality product.

REFERENCES

https://en.wikipedia.org/wiki/
http://www.softwaretestinghelp.com/

ABOUT THE AUTHOR

A Xoriant India Testing professional with 10 years of proven Quality and Test management experience in multiple domains including web, healthcare, mobile, telecom, client-server. Passion for quality and consistently working towards implementing new ideas and techniques to improve the overall quality management of projects.

To reach, email at: - priyanka.aggarwal@xoriant.com