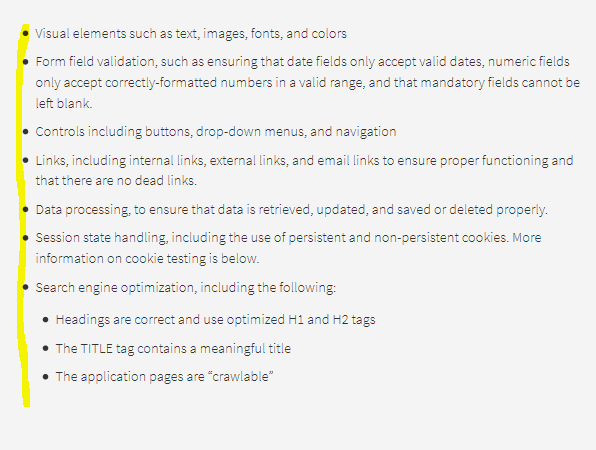
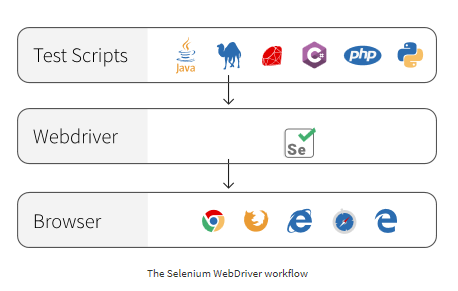
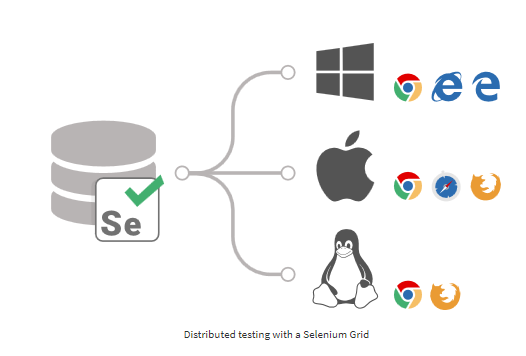
## Web application functional testing checklist

Functional testing addresses the following aspects of a web application. Elements to check on each page of the application include the following:



In addition to checking individual elements, be sure to validate a typical user’s path through the system, called the “happy path.” For instance, a hotel booking website “happy path” would include searching for hotel room availability, selecting a room, entering reservation and payment details, and completing the reservation. It is important to also test the “sad path” negative test cases, such as no availability on a given date, and “bad path” error conditions such as a request for a reservation for a date in the past.





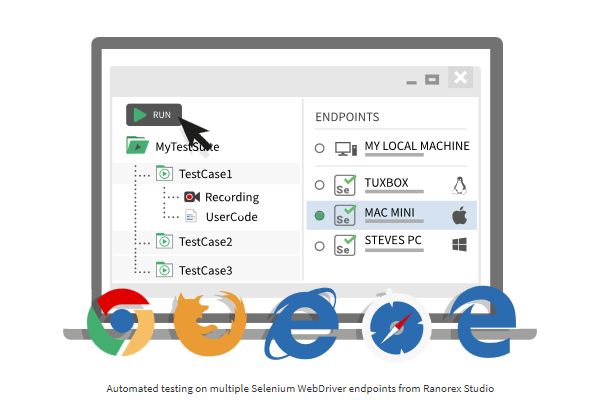
**Why use Selenium WebDriver for web testing?**

**Benefits**

* Selenium Webdriver is free and open-source
* The Selenium project is promoted by leaders in the web development and software testing industries
* It will support the emerging W3C WebDriver standard
* It offers the ability to run tests in parallel using Selenium Grids

**Challenges**

* Selenium Webdriver is complex to install and configure
* Building automated tests requires advanced programming knowledge, especially since there is no capture-and-replay functionality
* The Selenium IDE no longer works in Firefox version 55 and later, due to changes in the Firefox browser.
* There is no built-in test results reporting functionality

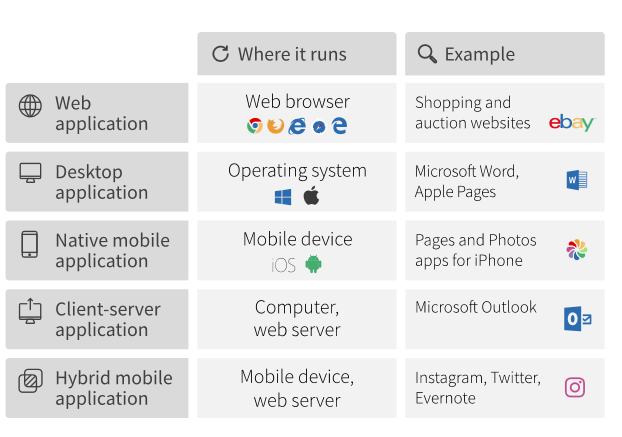


**How to get started with Selenium WebDriver**

Selenium WebDriver is built into the Ranorex core, so that you can leverage all of the power of Selenium WebDriver – along with Ranorex Studio’s tools for element identification, timeout handling, capture-and-replay codeless automation, full IDE, and built-in reporting. Ranorex Studio supports running tests in parallel using either a Selenium Grid or Ranorex’s own Parallel Runner. If you prefer to set up your own Selenium tests by hand, first select a programming language and IDE to use with Selenium Webdriver – such as C# or JavaScript with Microsoft Visual Studio, or Java with Eclipse. Then, download and install the [Selenium Server](http://www.seleniumhq.org/download/) and the Selenium Webdriver [client API](http://www.seleniumhq.org/download/) for your chosen programming language. Next, download and install your target browser, such as Chrome, on your test device. You will also need the browser-specific driver, such as the ChromeDriver. For a step-by-step tutorial on setting up WebDriver for Java with Eclipse, refer to the Selenium Setup Guide at [AutomationTestingHub](http://www.automationtestinghub.com/selenium-3/).

**Tip**

To learn more about Ranorex Studio’s deep integration with Selenium WebDriver, watch one of our webinars live or on-demand, or download a free 30-day trial of the full version of Ranorex Studio.



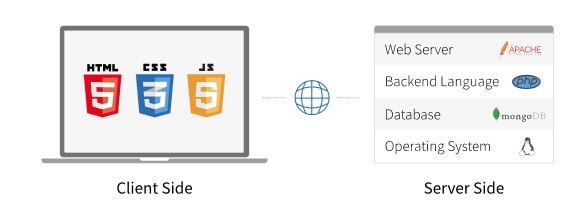
**Web application testing: terms to know**

Understanding the following terms will be useful as you get started with web application testing.

|  |  |
| --- | --- |
| HTML | Hypertext Markup Language. Along with CSS, HTML is the building block of a web page. HTML tags format the text on a page with headings, bullets, italics and bolding, active hyperlinks, and so forth. For example, the **h1** tag marks the top-level heading on a web page. |
| CSS | Cascading Style Sheet. CSS describes the colors, fonts, page margins and other layout elements on a web page. For example, the CSS may format the h1 tag with a color, font-size, font-weight, and border. |
| DOM | Document Object Model. When a browser loads a web page, it combines the HTML page content with the CSS style definitions to create the DOM and display it to a user. The DOM has a tree-like structure. Each element of the web page is a node in the tree structure. DOM nodes are defined by their relationship to other nodes, such as parent, child, or sibling. |
| HTTP | Hypertext Transfer Protocol. A communication protocol used to exchange information between browser, client application, and server. HTTPS is a secured version of HTTP. HTTP consists of methods, such as GET to retrieve data, and POST to submit data for processing. HTTP is stateless, meaning that any given request has no access to previous requests. Therefore, web developers have to use methods such as cookies to “remember” the state of a session, such as whether or not a user is logged in to the application. |
| JavaScript | An HTML page is static – meaning that it doesn’t change in response to user input. To make static web pages work more like desktop applications, programmers use JavaScript or a similar programming language to add responsive functionality such as clickable buttons. jQuery is a free, open-source library of JavaScript functions that speed up the process of web page development. |
| XML | eXtensible Markup Language. A browser and a server can exchange only text information. XML is used to represent data as text and exchange it between browser and server. |
| JSON | JavaScript Object Notation. JSON is similar to XML in that it is a way of exchanging data as text between a browser and a server. Where JSON differs from XML is that JSON objects are ready-to-use by JavaScript functions, where XML objects require an XML parser. |
| AJAX | Asynchronous JavaScript and XML. AJAX combines JavaScript (or a jQuery function) with either JSON or XML to exchange data with a server and update web page elements in the background, without needing to reload the entire page. |
| Cookie | A cookie is a small data file that stores information. There are several different types of cookies. For example, a session cookie lasts only as long as a user keeps a browser open and is deleted when the browser closes. By comparison, a persistent cookie lasts beyond a single browser session. A persistent cookie for a shopping site could keep track of items a user has placed in a shopping cart so that the user does not have to re-add them if the browser is closed before checking out. |

### Looking deeper: understanding web technologies

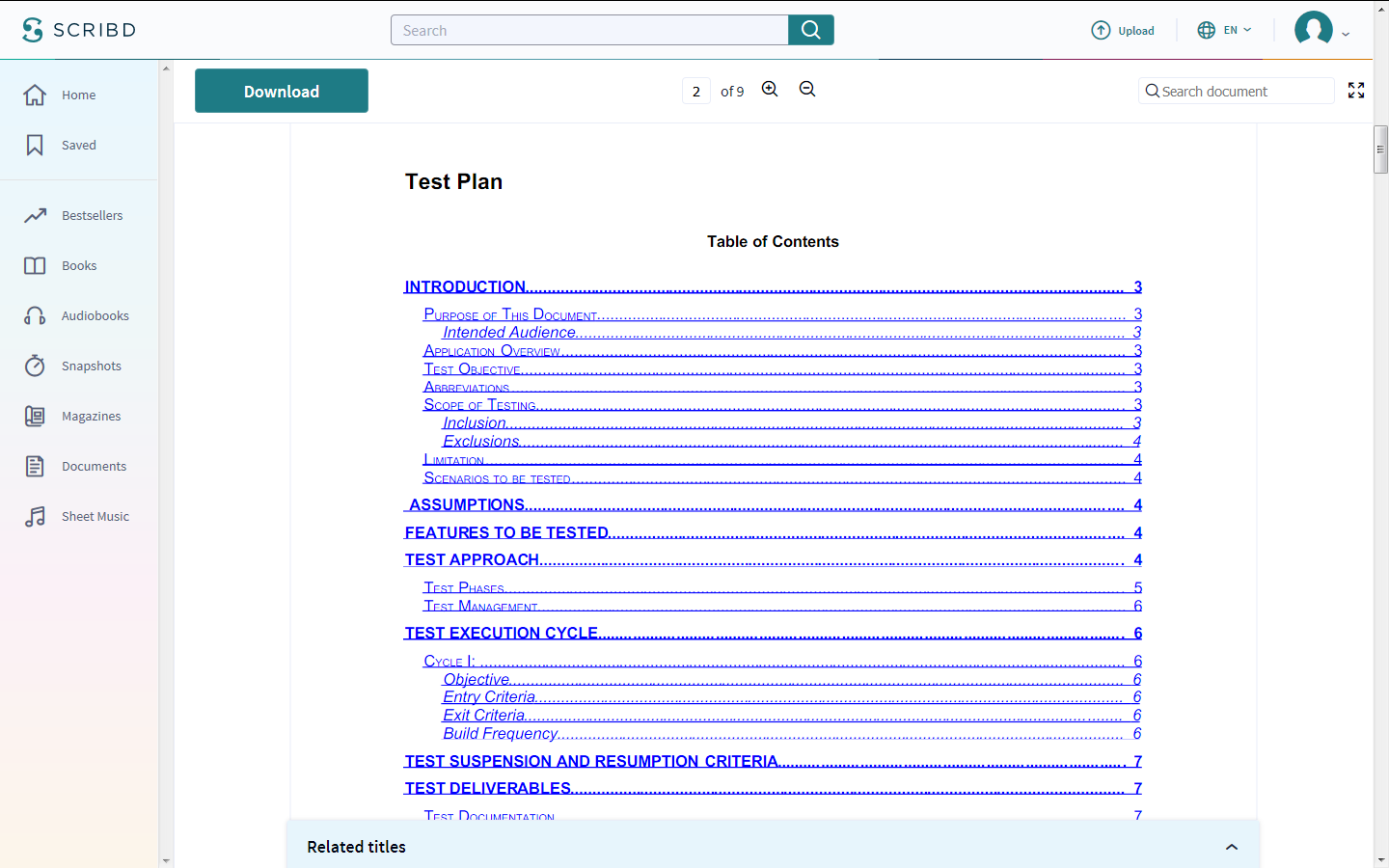
It is possible to test the UI of a web application without knowing the underlying architecture – this approach is called “black box” testing and is the focus of most of this article. However, understanding the web technologies that comprise your particular application will help you move beyond testing the UI to testing other aspects of the application as well, using “white box” testing techniques. The diagram below shows a web application divided into a “client side” and a “server side.” On the client side, HTML, CSS and JavaScript code make up the part of the application that presents the UI to the user. The client side uses HTTP requests to exchange information with the web server. Front-end unit testing validates the HTML, CSS and JavaScript code that makes up the part of the application that is visible to the user. There are several popular JavaScript frameworks available to web developers for front-end testing, including Mocha, Jasmine, and Karma.



The back-end application is usually written in languages such as JavaScript, Ruby, Python or PHP and is hosted on a web server such as the Apache HTTP web server. Data that the application needs is maintained in a back-end database – either a SQL database such as MySQL, Oracle or Microsoft SQL server, or a non-SQL database such as MongoDB. The final layer of the web application is the operating system that hosts the web server, such as Linux or Microsoft Windows Server. Together, these technologies are called the “web stack.” One of the most common and well-established web stacks is known by the acronym LAMP: a stack that includes the Linux operating system, Apache web server, MySQL database and a client-facing application written in PHP (or Python or Perl). Another common web stack and [competitor to the LAMP stack](https://www.infoworld.com/article/2937159/javascript/mean-vs-lamp-for-your-next-programming-project.html) is known by the acronym MEAN: which includes the no-SQL MongoDB database, Express.js for data flow, AngularJS for data presentation, and Node.js as the web server. JSON is used to format data throughout the stack. Because the MEAN stack uses JavaScript and JSON on both the front-end and back-end, the complexity of the stack is reduced. In addition, the MEAN stack is not tied to one operating system, as is the LAMP stack.

# Automation Test Plan

Test PlanAutomation Test Plan for <Project Name>



Test Plan

Introduction

Purpose of This Document

This document will be reviewed and accepted by the client (technical contact) and it specifically coversthe following areas:

1.The scope of testing

2.The various phases and types of testing required for product documentation

3.Description of the test environment

4.The test data requirements

5.Resources and schedules

6.The test management and process controls

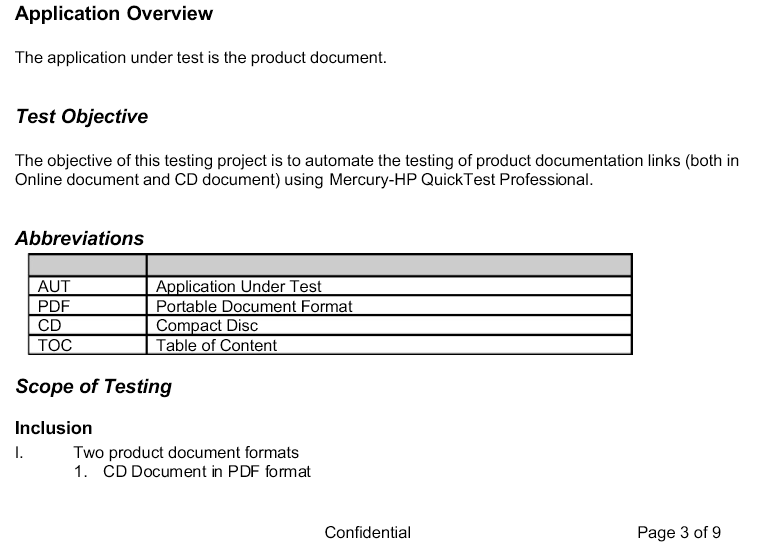
7.The test documentation to be produced for this project

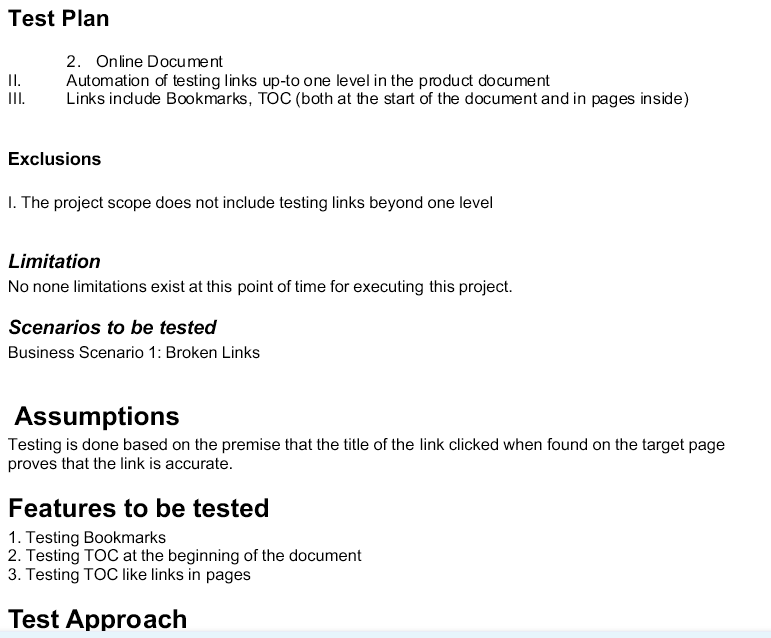
Intended Audience

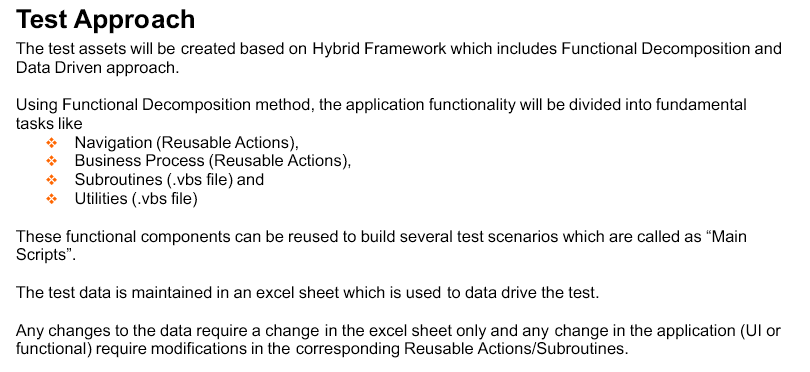
The test plan will be used to bring the entire team including the technical, administrative and businessteam members to the same level of understanding.This document along with any further updates will be sent to:

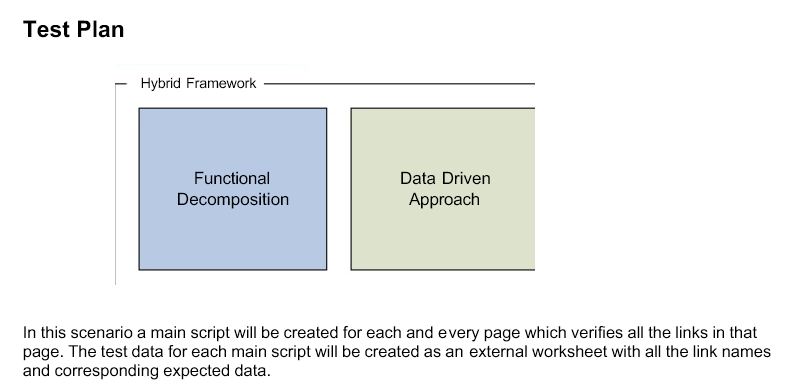
1.Client Technical Contacts

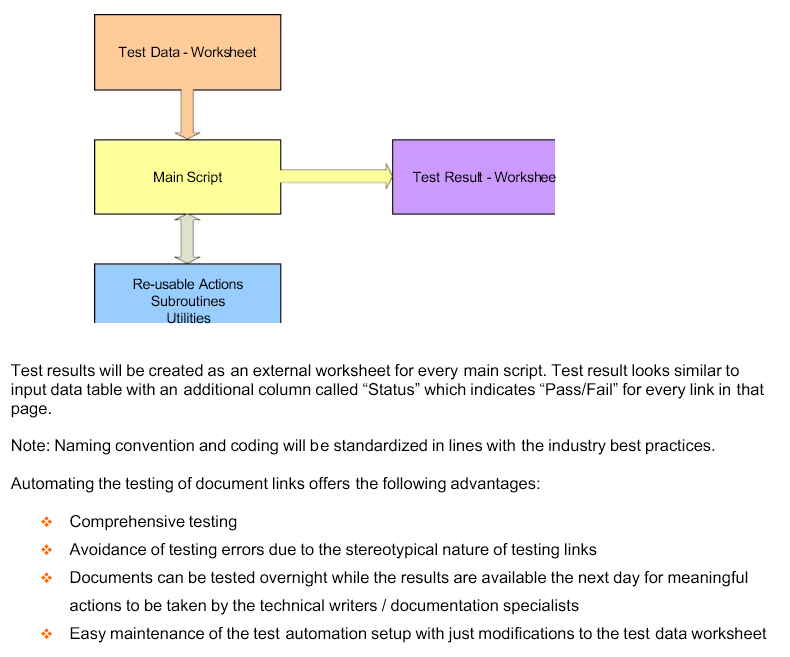
2.Test Automation Engineers



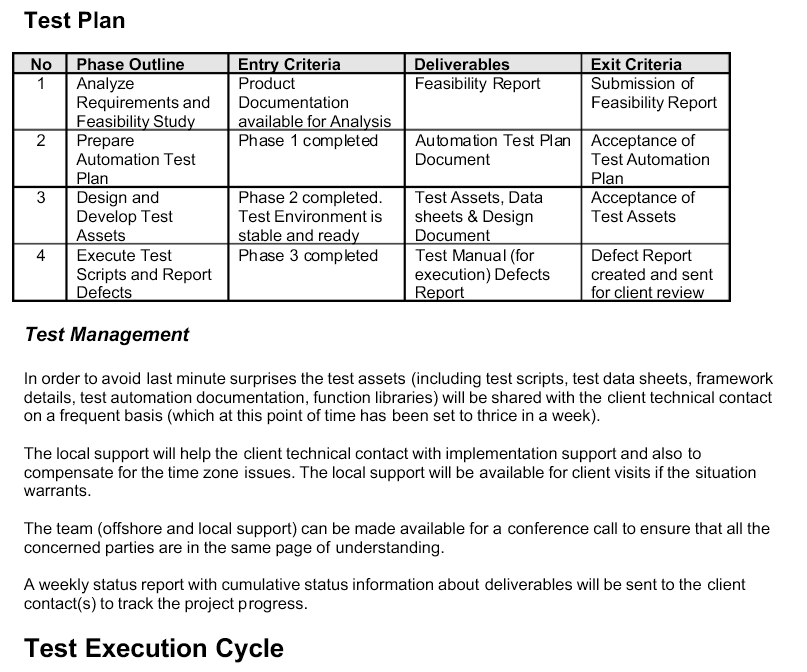


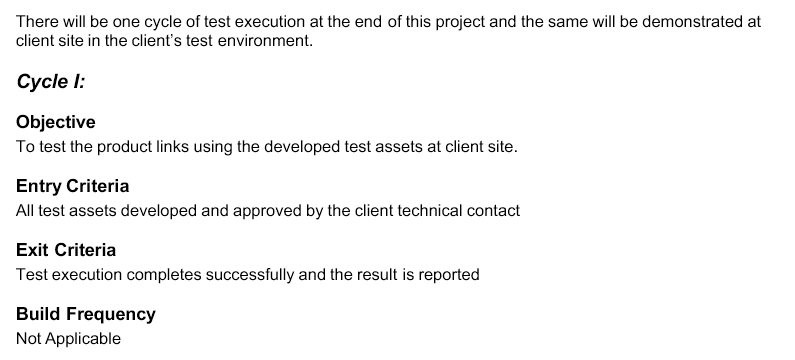


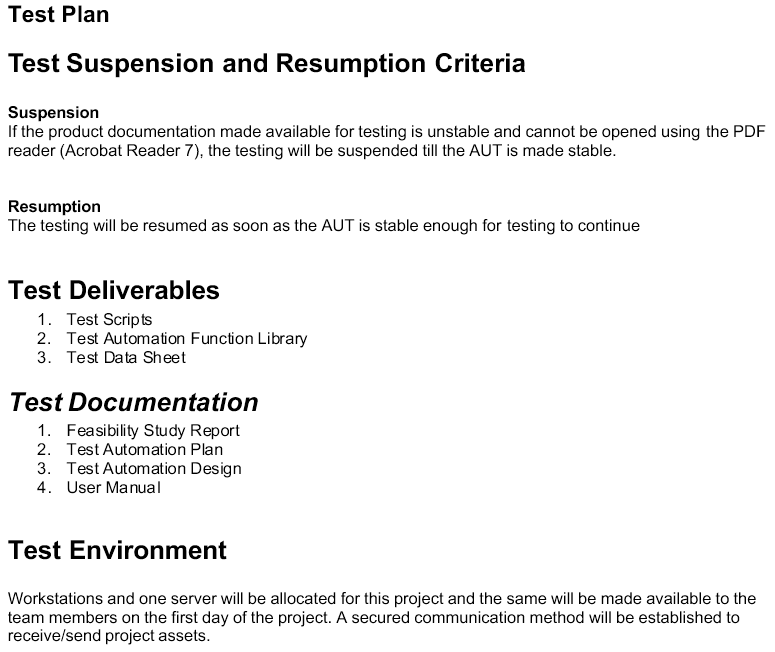


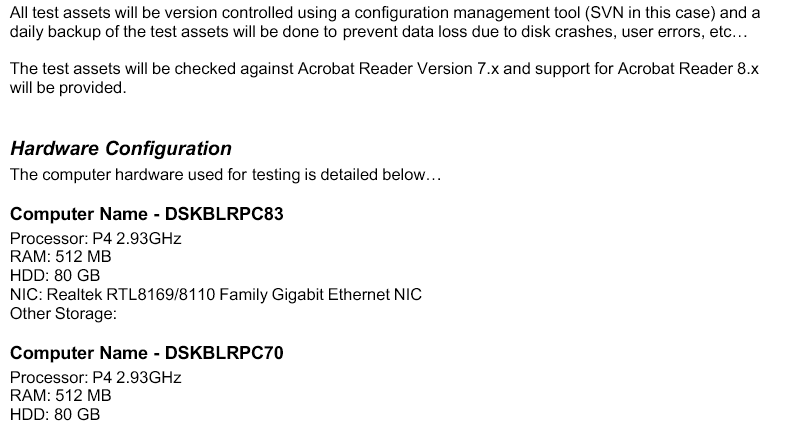


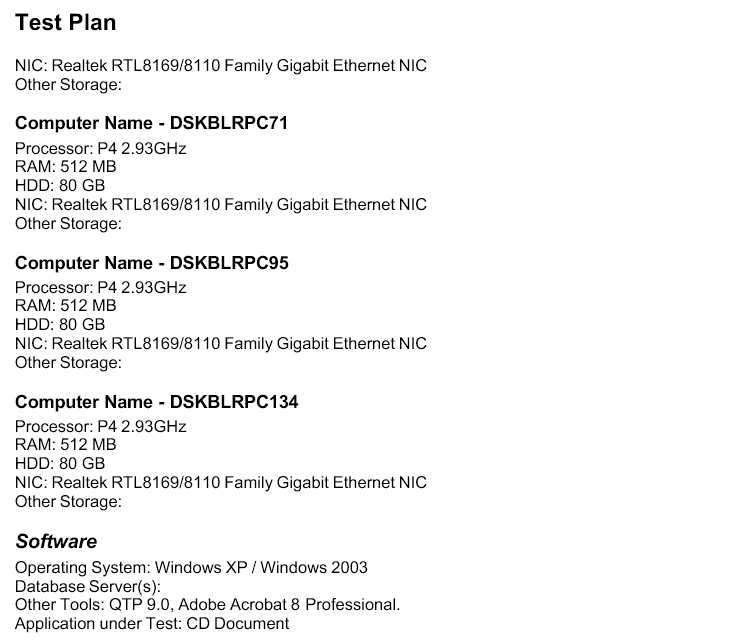


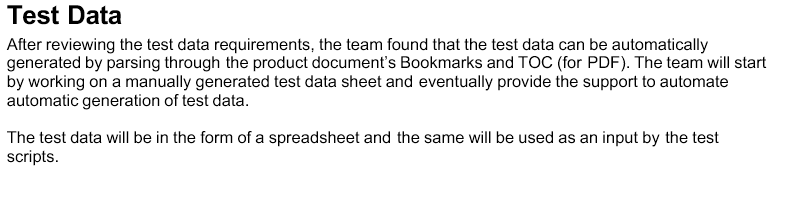


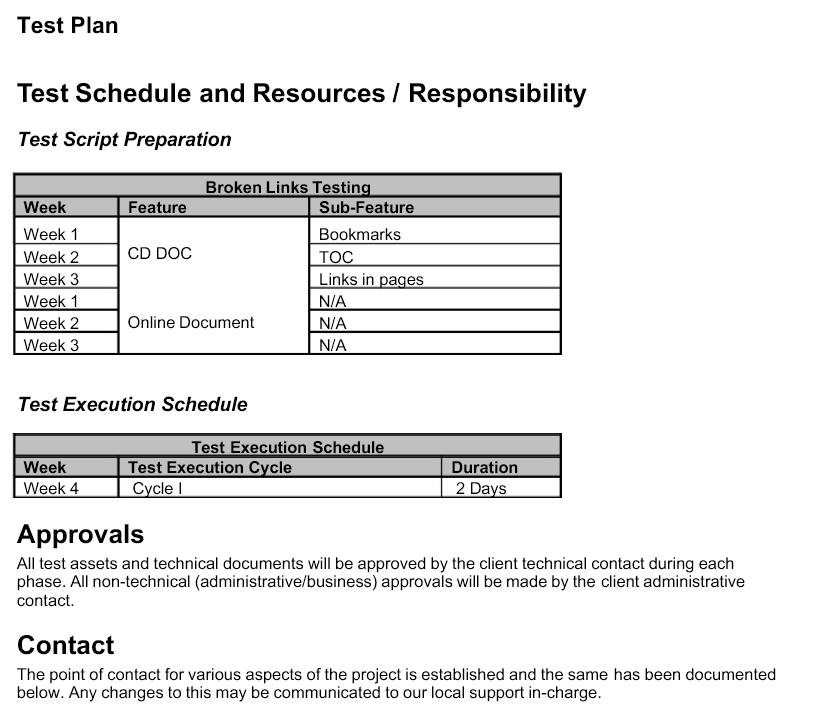


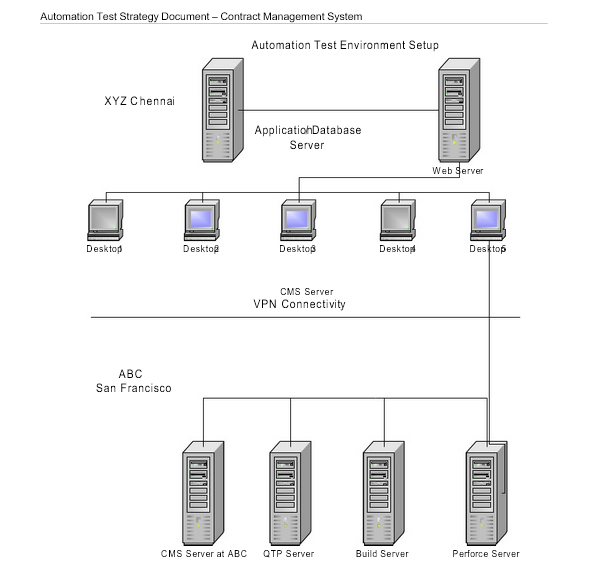












<https://www.scribd.com/document/56403046/Automation-Test-Plan>