**Team Kumquat**

**Software Testing Strategy**

**Product**

**“BLOG”**

**Version: 0.0.0.1** **Date: (04/24/2017)**

**Table of Contents**

[**1. Introduction**](#_30j0zll) **3**

[**1.1 Purpose**](#_1fob9te) **3**

[**1.2 Definitions, Acronyms, and Abbreviations.**](#_2et92p0) **3**

[**1.4 Overview**](#_3dy6vkm) **4**

[**2. The Overall Description**](#_1t3h5sf) **4**

[**2.3. Available Functionality**](#_i1tv376yelq9) **4**

[**2.4. Constraints and Requirements**](#_ri22cowlu2ou) **4**

[**2.5. UI**](#_477r4kdj245t) **5**

[**2.6. General functionality by views**](#_autgl2kqrktx) **5**

[**3. Validation of System Requirements and selected Test Cases**](#_qi123mtqc4ae) **6**

[**5. Test Report template**](#_npmlap6uqvrp) **9**

[**6. Manual Testing tasks**](#_6888xvix8csw) **9**

[**7. UI testing with tools tasks**](#_ku6bevs6w3ql) **9**

# **1. Introduction**

This article outlines the test strategy and cases covered by the QA team.

## **1.1 Purpose**

The following document targets the QA team responsible of verifying and validating the functionality of the “Blog” application. The document contains brief description how the software is validated and to what extend.  
  
The target of the QA team is to test the product to validate users can use it without encountering unhandled errors and input fields are validated. Test cases cover concurrency usage by users.  
  
The testing strategy does not cover system integration.

## **1.2 Definitions, Acronyms, and Abbreviations.**

MVC - **Model–view–controller** is a software architectural pattern for implementing user interfaces on computers. It divides a given application into three interconnected parts in order to separate internal representations of information from the ways that information is presented to and accepted from the user.The MVC design pattern decouples these major components allowing for efficient code reu[se](https://en.wikipedia.org/wiki/Code_reuse) and parallel development.

LocalDB - a lightweight version of the SQL Server Express Database Engine that is targeted for program development

MSSQL - **MS SQL** Server is a relational database management system (RDBMS) developed by Microsoft. This product is built for the basic function of storing retrieving data as required by other applications. It can be run either on the same computer or on another across a network.

VS - **Microsoft Visual Studio** is an integ[r](https://en.wikipedia.org/wiki/Integrated_development_environment)ated development environment (IDE) from Microsoft. It is used to develop computer programs for Microsoft Windows, as well as web sites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code.

Selenium WebDriver - designed to provide a simpler, more concise programming interface in addition to addressing some limitations in the Selenium-RC API. Selenium-WebDriver was developed to better support dynamic web pages where elements of a page may change without the page itself being reloaded. WebDriver’s goal is to supply a well-designed object-oriented API that provides improved support for modern advanced web-app testing problems.

NUnit - NUnit is a unit-testing framework for all .NET languages. Initially ported from JUnit, the current production release, version 3, has been completely rewritten with many new features and support for a wide range of .NET platforms.

.NET - a software framework developed by Microsoft that runs primarily on Microsoft Windows. It includes a large class library named Framework Class Library (FCL) and provides language interoperability (each language can use code written in other languages) across several programming languages.

To be continued with all products…

## 

## **1.4 Overview**

*The System Requirements are validated by decision tables. The test strategy is built on exploratory principle.  
It is suggested that the system is hosted accurately and users have the required browser access to the application. Navigation outside the side is not covered by the selected testing strategy.*

# **2. The Overall Description**

## **2.3. Available Functionality**

* 1. User
* Register with e-mail, Full name, password. Each must be verified.
* Login by email and password with option to remember the credentials
* Edit account - change password only;
* List articles
* Create article
* Edit own article
* Delete own article
* Read all articles
* Log off
  1. Administrator User
* Standard user
* List users
* Edit user account
* Delete user account
* Edit any article
* Delete any article

## **2.4. Constraints and Requirements**

* Only one user should be able to edit an article at a time
* Users edited or deleted by an admin should be logged off on submitting the change
* Multiple users should be able to login at a time
* Admin account is [admin@admin.com](mailto:admin@admin.com) password 123 (username Admin)

## **2.5. UI**

* Article/List
* Account/Register
* Manage/ChangePassword
* Account/Login
* Article/Create
* Article/Details/id
* Article/Edit/id
* Article/Delete/id
* Users/List
* Users/Edit/id
* Users/Delete/id

## **2.6. General functionality by views**

* Welcome screen Article/List - articles by all users
* Register/Login links or Create/welcome message/log off options listed on top depending on user is logged or not
* SOFTUNI logo leading to welcome screen
* Registration form Account/Register
* User screen =Welcome screen with displayed welcome message
* User edit account Manage
* User change password Manage/ChangePassword - status view is Manage?Messgae=XXXXX
* Log off goes back to Welcome screen without displayed welcome message
* Login Account/Login (remember me function) - leads to welcome screen with welcome message
* Create button for logged users - Article/Create
  + Cancel goes back to welcome screen with displayed welcome message
  + Create goes back to welcome screen with displayed welcome message and newly added article in the list (title, content, author must be displayed) \*author is not displayed, unless it is the admin user (bug)
* Clicking an article in the welcome screen goes to Article/Details/[article\_id] with Edit/Delete/Back options
* Edit/Delete operations on not owned articles bring 403 unhandled error (not authorized action)
* Back button goes back to welcome screen with welcome message
* Edit on owned article goes to Article/Edit/[article\_id]
  + Cancel goes back to Welcome screen with displayed welcome message
  + Edit saves changes and goes back to the welcome screen with welcome message
* Delete on owned article goes to Article/Delete/[article\_id]
  + Cancel goes to welcome screen with welcome message
  + Delete removes the article and goes back to welcome screen with welcome message

Administrator additional screens:

* On Article Edit/Delete can make the changes without the 403 authorization error
* Admin link next to the welcome message leading to Users/List
* Edit User goes to User/Edit/[user\_id\_as\_guid]
* Options to make the user admin, change email, full name, password (no cancel or go back, saved change are reflected at Users/List)
* Delete user goes to User/Delete/[user\_id\_as\_guid]
  + Cancel goes back to User/List
  + Delete saves changes and goes back to User/List

## 3. Validation of System Requirements and selected Test Cases

Validation of authorization:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Test cases | | | | | | | | | |
|  | Case1 | Case2 | Case3 | Case4 | Case5 | Case6 | Case7 | Case8 | Case9 | Case10 |
| Inputs | List Articles | List Users | Create Article | Delete/ Update Own Article | Delete/  Update Foreign Article | Preview Article Details | Preview User Details | Delete  Update  User | Update password | Register/  Login view |
| Not logged user | Y | N | N | N | N | Y | N | N | N | Y |
| Logged User | Y | N | Y | Y | N | Y | N | N | Y | N |
| Logged Admin | Y | Y | Y | Y | Y | Y | Y | Y | Y | N |

* Case2 and Case7 can be merged
* Case1 and Case6 can be merged

Special cases:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Test Cases | | | | |
| Conditions | User2 Edits Article1 | User2 Deletes Article1 | Admin1 Edits User1 | Admin1 Deletes User1 | Admin1 Deletes Admin2 |
| User1 reads article1 | Keep reading the last visible info | Keep reading the last visible info | Log off->  Register screen | Log off->  Register screen | N/A |
| Admin2 edits Article1 | Last submitted changes are used | Remove article | N/A | N/A | IIf Edit changes are submitted to the article, accept and log off Admin2 ->Register screen |

The above table contains rules not determined by the system requirements.

Input validation:

* Register screen for User|User Edit screen for Admin - Full name cannot be invalid

|  |  |  |  |
| --- | --- | --- | --- |
|  | Test Cases | | |
| Condition | Case1 | Case2 | Case3 |
| Invalid email | Y | N | N |
| Invalid full name | - | - | - |
| Not matching passwords | Y | Y | N |
| Output |  | | |
| Registration | No | No | Yes |

* Login screen

|  |  |  |  |
| --- | --- | --- | --- |
|  | Test Cases | | |
| Condition | Case1 | Case2 | Case3 |
| Invalid email | Y | N | N |
| Invalid password | Y | Y | N |
| Remember me? | - | - | - |
| Output |  | | |
| Login success | No | No | Yes |

* ChangePassword screen

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Test Cases | | | | | | | |
| Condition | Case1 | | | | Case2 | Case3 | Case4 | |
| Incorrect Old password | y | y | y | y | n | n | n | n |
| Not matching new passwords | y | y | n | n | n | n | y | y |
| The same old/new password | y | n | y | n | n | y | n | y |
| Output |  | | | | | | | |
| Change success | No | | | | Yes | No | No | No |

Greyed input values can be ignored and test cases can be merged.

* Create/Edit article

|  |  |  |  |
| --- | --- | --- | --- |
|  | Test Cases | | |
| Condition | Case1 | Case2 | Case3 |
| Empty Title | Y | N | N |
| Empty Content | Y | Y | N |
| Output |  | | |
| Save success | No | No | Yes |

* Edit User

## 5. Test Report template

The template which is used for reporting issues is in a format:

1. URL
2. Steps to reproduce
3. Type of user - standard or admin
4. Type of testing - manual or automated UI test
5. Error message and exception stacktrace
6. Handled or unhandled by cod Yes/No
7. Requires further development yes/No

## 6. Manual Testing tasks

* Go through point *2.6 general functionality by views* and manually test the functionality in IE11, Chrome, Firefox. Verify the UI is displayed correctly and check for unhandled errors.
* Use two browser windows to create and log with different users, where the first one is an administrator. Test if deleting the second user is properly handled without errors on screen
* Use two browser windows to create and log with different users, where the first one is an administrator. Test if deleting an article created by the second user is properly handled without errors on screen.

Report each issue in a DOCX file with the format given in point 5. Test Report template.

## 7. UI testing with tools tasks

* Generate a model describing the screens suggested in *point 2.5 UI*
* *Create input data based on the decision tables in point 3. Validation of system Requirements.*
* *Gather the expected results for each test from point 3. Validation of system requirements*
* *Create tests checking the validation of input fields for each screen having such specified in point 2.5. UI*
* *Create tests covering the manual testing scenarios*

Report each issue in a TXT file with the format given in point 5. Test Report template. The file should be generated automatically by the run tests.