## Purpose

This document explains the various testing activities performed as part of Testing **CRUD Articles functionality** of ‘BBlog" social blogging site’ application.

## Test Strategy

### Test Scope

#### In Scope

Functional Testing for the following functions in Scope of Testing:

* Create Article
* Read Article
* Update Article
* Delete Article

#### Out of scope

* Functional testing of other general functionalities
* CR\*D Comments on articles (no updating required)
* GET and display paginated lists of articles
* Favorite articles
* Follow other users

#### Items not tested

* Performance Testing was not done for this application.
* Compatibility testing was not performed across different browsers
* DB verification was not done for this application
* API testing was not performed using an API testing tool as the UIs are available

### 2.2 Test case selection Criteria

* requirements-based criteria
* data based criteria

### Positive Test cases

* **C1reate** - Start at the home page, click the New Article button, input form values, click the Publish Article button. - **Automated**
* **R1ead** - Start at the home page, go to tab your feed, select the Article, click on Read more link. -
* **R2ead** - Start at the home page, go to tab Global feed, select the Article, click on Read more button - **Automated**
* **R3ead-** Start at the home page, select tag, go to matrix tab, select the Article, click on Read more button - **Automated**
* **R4ead** - Start at the home page, select user, go to user profile page, select the Article under my posts, click on Read more button – **Automation covered with U1pdate and R2ead tests**
* **R5ead** - Start at the home page, select user, go to my profile page, select the Article favorited posts, click on Read more button
* **U1pdate** - Start at the home page, go to my profile, select article under My posts, click on Read more, Click on Edit button, change form values, click the save button. **Automated**
* **D1elete** - Start at the home page, go to my profile, select article under My posts, click on Read more, click on delete button. - **Automated**

### Negative test cases

* **C2reate** - Start at the home page, click the New Article button, input empty form values, click the Publish Article button. **- Not automated as the validations were not handled**
* **U2pdate** - Start at the home page, go to my profile, select article under My posts, click on Read more, Click on Edit button, input empty form values, click the save button. **- Not automated as the validations were not handled**
* **U3pdate** - Start at the home page, go to tab Global feed, select the Article created by another user, click on Read more button, Edit Article button should not be displayed
* **D2elete** - Start at the home page, go to tab Global feed, select the Article created by another user, click on Read more button, Delete Article button should not be displayed

### Automation candidates’ selection Criteria

1.Test cases which verify core features of the product which has a repetitive nature(Tests to Verify the happy path):

* Test method to test one operation of each: Create, Read, Update, and Delete
* The creating and updating tests also testing Read in article page, because we are retrieving the user for our assertions. Therefore, we can reduce the number of test cases to be automated.
* E2E tests - Web UI testing validates that all components of the web application are well-connected. Automated testing flow should cover most of the core functionalities of the system

2.Functionalities which are more visible to the users

* User can select article for reding under several tabs. we need to cover them as well: Your Feed, Global Feed, My posts, Favorite posts
* Filter Articles by tags

3.some negative scenarios with our CRUD testing which undergoes with core functionalities of the application

4.Flows which contains with defects were not automated and removed from regression suite

Assumption:

* We could assume that our user will always be present in the database
* Already created articles by other users will be present in the database

1. Risks

* Data integrity was not properly handled – incomplete validations the data at the page level when user enters the data
* DB is not accessible and DB verification not done
* Lacking requirement, hard to think of how the expected outcome should be

## Recommendation

* The client-side of the application is not always responsible for data integrity. Thus, it is necessary to validate its integrity through the APIs, the Database layer, and the UI
* Access permission to DB to verify DB changes when performing CRUD operations to verify that data changes from front-end are represented in DB accordingly.
* In case UI s are not yet completed we can verify the API, to successful API Test, you need to have the correct tools such as Postman