Testing for Mainteny Login page

The Login feature is In-Scope for Functional, Security, Performance, and Interface Testing.

***Assumption:***

***Test URL is accessible and hosted server is up and running most of the time.***

***The tester can start and stop the server.***

***No new feature will be added to the test URL***

# Functional Testing

## Required Fields validation

## Testable Scenarios

* Verify that if user missing with email or password, login button is disable.
* Verify that when user fills the required field login button is enabled.
* Verify that when user enters correct password and email and click [Login] button should be taken to dashboard.

## Email validation

## Testable Scenarios

* Verify the Email text field that has an Email address without @ symbol
* Verify all the valid emails.
* Verify if user enters non-existing email, get error message

## Password validation

## Testable Scenarios

* Verify that user can see the password which he types when click on eye icon.
* Verify when user enter wrong password and try to click login, user get error.

## Link validation

## Testable Scenarios

* Verify that when the user clicks on [Try Demo] button in left pane should be taken to demo website.
* Verify Forget Password opens page to reset password.
* Verify that when user clicks on [Try Demo] button in top should be taken to demo website.
* Verify that when user clicks on the Logo, should be taken to <https://mainteny.com/> page.

# Non-Functional Testing

# 1. Performance Testing / Security Testing

## Testable Scenarios

* Verify that when multiple users are trying to login account, each request is handled gracefully, the web app should not crash.
* Verify that response time for each action performed by the user should be in acceptable deviation.
* Verify that same user can not login from two different browser at same time.
* Verify that user is able to login in different networks i.e. WIFI, Mobile, LAN etc.
* Verify that when sever is down user trying to login should get appropriate error message.

# 2. Usability Testing

## Testable Scenarios

• Verify that the Menus, Links, or buttons on the application move to the correct pages or windows.

• Verify for all the spelling and grammatical errors present on the page.

# 3. Interface Testing

## Testable Scenarios

• Verifies that communication towards all the three servers – The Web, Application, and Database Server is working fine.

• Verify if any request interrupts in-between then how the application is responding. Handle any error from the web or database server to the application server and display the correct error message to the user.

**There are a few open-ended questions**

1. What will happen to login request if the server gets restarted?
2. Are we also doing any HA (high availability) testing?
3. Is the data getting written directly to the database?
4. Will there be any warning if a memory swap or disk failure happens?

# Propose Automation Tool

It would be good to use **cypress** for automation here are key points:

1. Zero configuration to get started.

2. Low learning curve

3. Automatic waiting

4. Control of network traffic

5. Support various automation test i.e., API, front-end, end - end test, visual regression tests

6. No need to choose framework i.e. (Mocha, Jasmine, test NG)

7. No need to choose assertion library (Chai, expects)

8. No additional installation (Selenium, Java, etc.)