Tasks

TASK 1 – EXPLORATORY TESTS

Go to the UI Test Automation website and identify manually any issues and briefly list them including:
• Description – a brief description of the issue identified.
• Severity – determine whether High, Medium or Low with a short explanation justifying why.

DESCRIPTION	SEVERITY
Emails with valid format prompt the form to provide an error message. - I.e. "john.halo123@gmail.com" prompts an error message stating "Invalid email address.".	**High -** Many users will have an email of alphanumeric nature, and so this will create a lot of frustration and an impassable roadblock for many users, and undermine the validity of the form.
Form navigation breaks with multiple clicks of "Next" Button. - user double-clicks the "Next" button, the form proceeds to the incorrect page. I.e. If the user is on page 1 with email registration, double clicking next takes the user to the contact form whilst the navigation number is still 2, and completely skips contact information.	**High** - This issue severely impacts the user experience by causing confusion, while also breaking the flow of the form registration process since the user is able to completely bypass the contact form stage.
Contact form allows users to submit the required fields denoted with asterisks with incorrect input. I.e. A single whitespace character, special characters, numeric characters etc.	**High** - Allowing invalid input as valid in required fields undermines the integrity of the form and can result in incomplete or invalid submissions.
Payment form card holder name allows invalid input (Whitespace characters, special characters, numeric characters etc.)	**High -** Again, passing invalid input as valid undermines the integrity of the form and results in invalid submissions.
Entering a card number that is too short or too long does not trigger an error message. (I.e. 1234 or 12345678912345678912345)	**High -** The payment page should enforce a valid card number. Passing invalid input undermines integrity of the form and results in invalid submissions.
The contact form's postcode field does not properly validate input.	**High -** The postcode form field allows illegal non-numeric characters, instead of checking for valid postcodes.
Valid length credit card numbers that do not pass Luhn's algorithm (such as 1111 1111 1111 1111) are allowed.	**High -** The form should check with Luhn's Algorithm and reject invalid but correct credit card number forms.

Invalid (past, or lengthy expiry date years are allowed)	**High -** Invalid expiry years are accepted, this leads to users being able to submit incorrect or expired card details and overall invalid submissions.
Card CVV allows input of values that are not 3 or 4 numeric characters long. (i.e. 3, 33, or 33333)	**High -** Invalid CVV values could result in the payment process failing, and may create frustration for the user.
Victoria is not listed in the dropdown menu for Australian states.	**High -** Users, primarily those residing within Victoria, will not be able to select it as their state or territory creating frustration, but more importantly, impeding them from completing the form and undermining its integrity.
On the contact page, if first name is filled and last name is not, there is no error thrown for last name field not being filled. There should be an error thrown for last name regardless if first name is filled or not.	**High -** Users should not be able to move forward with the form if they have not filled out the sufficient data. This will cause problems later down the line, and allows the user to bypass parts of the form with invalid input.
The system does not validate email addresses case-insensitively during registration. As a result, entering a pre-registered email with altered capitalization (e.g., 'Adam@Orikan.com' vs. 'adam@orikan.com') bypasses the 'user already exists' validation.	**Medium -** While the issue could confuse users and potentially cause duplicate accounts, it's unlikely to be immediately visible to users unless they notice the problem.
Email address already registered not indicated until registration form is complete.	**Medium** - Users may be spending exorbitant amounts of time on the login form inputting form details. Lack of immediate feedback on pre-existing emails creates frustration.
Partial page progression occurs even though an error occurs. - Occurs on each stage of the registration process. - Issue does not stack, meaning the number does not progress until the user is on the page that is displayed with the navigation numbers.	**Medium** - Users may notice that the form is not reliable may undermine the overall reliability of the website, and furthermore the brand.
Email submission does not handle leading or trailing spaces.	**Medium** - This issue can cause possible complications when verifying validity of the email at a later time. The form should automatically trim whitespace from the input or provide an error message telling the user “Please remove any extra spaces from the email address".

Preferred full name is populated with two whitespaces between the first name and last name even if middle name is not provided.	**Low -** The issue does not break the form and the preferred full name can be remedied by the user themselves, however it is still an issue that compromises the integrity of the overall form.
Tasmania is listed twice in the dropdown menu for Australian states.	**Low -** This issue does not prevent the form from being submitted. However, it creates unnecessary confusion and reduces user experience.

TASK 2 – AUTOMATED TESTS

Create an automated test suite in a testing tool of your choice (e.g. Selenium, Cucumber, Cypress, Playwright, SoapUI, etc.) that:

- automatically identifies the issues that were manually identified in Task 1
- automatically verifies currently working functionality in case it is later broken

Briefly list the test cases that you have included in your test suite.

TEST CASES
Test Suite: emailRegistration.test.js **Description:** This test.js file contains test cases which test the email registration page of the form. **Test Cases:** • 1: **Test case 1 POSITIVE CASE:** Form submission test case should submit. • 2: **Test case 2 POSITIVE CASE:** Form submission with alphanumeric email. • 3: **Test case 3 POSITIVE CASE:** Form submission with an alternate email. • 4: **Test case 4 NEGATIVE CASE:** Form submission should throw an error message under the email field because there is no input. • 5: **Test case 5 NEGATIVE CASE:** Form submission should throw an error message for the password field since it is not filled. • 6: **Test case 6 NEGATIVE CASE:** Form submission should throw an error message for the confirmPassword field since it is not filled. • 7: **Test case 7 NEGATIVE CASE:** Form submission should throw an error message since the email is already registered. • 8: **Test case 8 NEGATIVE CASE:** Test for case insensitivity in the system. • 9: **Test case 9 NEGATIVE CASE:** User should NOT be able to click on the next button after clicking it once, so the button must be disabled after the first click. • 10: **Test case 10 NEGATIVE CASE:** When the user clicks the next button on an unsuccessful registration, the page indicator should not fill in the next page as though the user has moved on.
Test Suite: contactRegistration.test.js **Description:** This test.js file contains test cases which test the contact registration page of

the form.

- 1: **Test case 1 POSITIVE CASE:** This test case fills out all required fields with valid inputs.
- 2: **Test case 2 NEGATIVE CASE:** This test case fills out all required fields with one whitespace character and checks if the site has not moved on to the next page.
- 3: **Test case 3 NEGATIVE CASE**: This test case fills out all required fields with invalid special characters and checks if the site has not moved on to the next page.
- 4: **Test case 4 NEGATIVE CASE:** This test case does not fill out required fields and checks if error messages are thrown for each field.
- 5: **Test case 5 NEGATIVE CASE:** This test case does not fill out first and last name and checks if error messages are thrown for both.
- 6: **Test case 6 NEGATIVE CASE:** This test case DOES NOT fill out first name but fills out the last name and checks if error messages are thrown for only the first name.
- 7: **Test case 7 NEGATIVE CASE**: This test case fills out the first name but DOES NOT fill out the last name and checks if error messages are thrown for only the last name.
- 8: **Test case 8 NEGATIVE CASE:** This test fills out first and last name and checks if there is only one space between first and last name in the automatically filled out preferred full name field.
- 9: **Test case 9 NEGATIVE CASE:** This test fills out postcode with value of length 1 and checks if an error is thrown.
- 10: **Test case 10 NEGATIVE CASE:** This test fills out postcode with value of length 2 and checks if an error is thrown.
- 11: **Test case 11 NEGATIVE CASE:** This test fills out postcode with value of length 12 and checks if an error is thrown.
- 12: **Test case 12 NEGATIVE CASE:** This test fills out city with a numerical character and checks if an error is thrown.

Test Suite: paymentRegistration.test.js
Description: This test.js file contains test cases which test the card payment details registration page of the form.
Test Cases:

- **Test Case 1 POSITIVE case:** Fill all payment fields with valid visa input and check if the next page has loaded
- **Test Case 2 POSITIVE case:** Fill all payment fields with valid mastercard input and check if the next page has loaded
- **Test Case 3 NEGATIVE case:** This test does not fill any of the required fields and checks if error messages are thrown for each
- **Test Case 4 NEGATIVE case:** This test fills the card holder name with invalid characters and checks for a corresponding error message
- **Test Case 5 NEGATIVE case:** This test fills the card holder name with invalid NUMERIC characters and checks for a corresponding error message
- **Test Case 6 NEGATIVE case:** This test fills the card number field with input of invalid short length and checks for a corresponding error message
- **Test Case 7 NEGATIVE case:** This test fills the card number field with input of invalid enormous length and checks for a corresponding error message
- **Test Case 8 NEGATIVE case:** This test fills the card number field with input of valid length but is not valid according to Luhn's algorithm and checks for a corresponding error message
- **Test Case 9 NEGATIVE case:** This test fills the card CVV field with input of invalid length 1 and checks for corresponding error message
- **Test Case 10 NEGATIVE case:** This test fills the card CVV field with input of

invalid length 2 and checks for corresponding error message
- **Test Case 11 NEGATIVE case:** This test fills the card CVV field with input of invalid length >3 and checks for corresponding error message
- **Test Case 12 NEGATIVE case:** This test fills the card expiry year field with input of past year and checks for corresponding error message
- **Test Case 13 NEGATIVE case:** This test fills the card expiry year field with input of year too far into the future and checks for corresponding error message

Test Suite: paymentRegistration.test.js
Description: This test.js file contains test cases which test the terms and conditions page of the form.
Test Cases:
- **Test Case 1 POSITIVE case:** Scroll to the bottom of the terms and conditions, click the agree button and submit the form
- **Test Case 2 NEGATIVE case:** Do not scroll to the bottom of terms and conditions, click the submit button and check if error is thrown
- **Test Case 3 NEGATIVE case:** Scroll to the bottom of terms and conditions, click the submit button without clicking agree and check if error is thrown

TASK 3 – DEVELOPER FEEDBACK

What feedback, if any, would you provide to the developers that would assist you in automating your test suite?

DEVELOPER FEEDBACK
It would assist in the testing process if the dev team could ensure that all interactive elements have unique identifiers. It would be preferable for no reliance on attributes like CSS selectors.
The progress indicator (circle with numbers) changes depending on where the user navigates throughout the registration form process. However, there is no clear attribute to indicate when the circle shape is filled with a colour denoting the user's presence on that page. For easier test automation, it would be helpful to add a clear state indicator.
During test execution, several tests frequently fail due to timeouts. To improve the reliability of automated tests, it would be helpful to optimize page load times. Currently, to circumvent this issue, a playwright.config.js file was created to increase timeout for tests and hooks to 8000ms.

TASK 4 – SHARE VIA GITHUB

The below user account has been added as a collaborator so that the test suite can be reviewed.

DETAILS	
Collaborator Username	mhenderson-orikan
Collaborator Email	marc.henderson@orikan.com
GitHub repository URL	https://github.com/RayCau/orikan-ui-automation-test
GitHub Username	RayCau
Chosen Testing Tool	Playwright