**Unit Test Case Documentation**

**AML Referral Component**

**Test Purpose:** Ensure the component is created successfully.  
**Test Description:**  
This unit test verifies that the component is created correctly when the component's constructor is called. It checks if the component instance is truthy (i.e., it exists and is properly initialized).

| **#** | **Test Case For** | **Scenario** | **Input** | **Output** | **HTTP Response** |
| --- | --- | --- | --- | --- | --- |
| 1.1 | Verify component creation | Component is created successfully | - | Component instance is exists and is properly initialized. | N/A |

**Test Purpose:** Verify that the form controls in the referralForm are initialized with the correct enabled/disabled state.  
**Test Description:**  
This unit test ensures that the form controls in the referralForm are initialized correctly, specifically that the appropriate form controls are enabled or disabled based on the expected behaviour. It checks the state (enabled/disabled) of each individual form control in the form.

| **#** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.2 | Verify form controls state | Form controls initialized successfully | - | Each form control in referralForm is set to the expected state (enabled or disabled). |

**Test Purpose:** Verify that the component correctly handles the case when account\_no is not present in the query parameters.  
**Test Description:**  
This unit test ensures that when the account\_no is missing from the query parameters, the component does not trigger any methods that require it during page loading. It also verifies that the form control for account\_no remains enabled and unaffected when no account\_no is present in the query parameters.

| **#** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.3 | Handle missing account\_no in query parameters | account\_no is missing; no dependent methods called, and form control remains enabled | Simulated absence of account\_no in query parameters using mockActivatedRoute.queryParamMap | - Methods like getAccountDetailsByAccountNo are not called.  - account\_no form control remains enabled. |

**Test Purpose:** Ensure getAccountDetailsByAccountNo is called during ngOnInit when account\_no is provided in the URL, and check the form control state and visibility behavior.  
**Test Description:**  
This unit test ensures that the getAccountDetailsByAccountNo method is invoked during the ngOnInit lifecycle hook if account\_no is present in the URL. It also verifies the following behaviors:

* The account\_no form control is disabled after ngOnInit.
* The isVisible property is set to true, ensuring that the Master RepId, Client Name, and Client SSN are displayed appropriately.

| **#** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.4 | Verify getAccountDetailsByAccountNo is called on ngOnInit | account\_no is provided in the URL, and methods are invoked properly | Simulated account\_no in query parameters using mockActivatedRoute.queryParamMap | - getAccountDetailsByAccountNo is called.  - account\_no form control is disabled.  - isVisible is set to true. |

**Test Purpose:** Ensure getReportersDetails and getTransactionType methods are called when the component initializes (ngOnInit).  
**Test Description:**  
This unit test verifies that the getReportersDetails and getTransactionType methods are invoked during the ngOnInit lifecycle hook of the component.

| **#** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.5 | Verify getReportersDetails and getTransactionType are called | Component is initialized (ngOnInit) | - | getReportersDetails and getTransactionType methods are called successfully. |

**Test Purpose:** Ensure the form is correctly populated with reporter details when the API call succeeds.  
**Test Description:**  
This unit test verifies that when the getReporterDetails method successfully fetches reporter details from the API, the form controls are updated with the corresponding values from the API response.

| **#** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.6 | Verify form is updated with reporter details | Successful API response for reporter details | Mock API response: { "reporter\_first\_name": "DevInt Vaishali", "reporter\_last\_name": "Zachrich", "reporter\_email": "vaishali.zachrich@lpl.com" } | Form controls (reporter\_first\_name, reporter\_last\_name, reporter\_email) updated with API data. |

**Test Purpose:** Ensure that the appropriate message is logged when no reporter details are returned from the API.  
**Test Description:**  
This unit test verifies that when the getReporterDetails method is called and the API returns no reporter details (i.e., a null response), the message "Reporter Details not found" is logged to the console.

| **#** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.7 | Verify logging when no reporter details are returned | No reporter details returned from the API | Mock API response: null | Console logs "Reporter Details not found". |

**Test Purpose:**  
Ensure that the appropriate message is logged when no reporter details are returned from the API.

**Test Description:**  
This unit test verifies that when the getReporterDetails method is called and the API returns no reporter details (i.e., a null response), the message **"Reporter Details not found"** is logged to the console.

| **#** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.8 | Verify logging when no reporter details are returned | No reporter details returned from the API | Mock API response: null | Console logs **"Reporter Details not found".** |

**Test Purpose:**  
Ensure that the reporter details error is correctly handled and logged when the API call fails.

**Test Description:**  
This unit test verifies that the component appropriately manages errors during the getReporterDetails API call. It simulates an error scenario by mocking the API service to reject with an error (e.g., mockError) and ensures that the error is logged to the console with the expected message.

| **#** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.9 | Verify logging of error when API call fails | API call fails during execution | Mock error: mockError | Console logs **"Error Occurred!"** and the error details. |

**Test Purpose:**  
Ensure that the getAccountDetailsByAccountNo method fetches account details from the API and correctly populates form controls when the API call succeeds.

**Test Description:**  
This unit test verifies that the component handles a successful response from the getAccountDetails API correctly. It ensures the account details (client\_name, client\_ssn, master\_rep\_id) are fetched and populated into their respective form controls as expected.

| **#** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.10 | Verify form controls are populated after successful fetch | Successful API response for account details | Mock API response: { "client\_name": "John Doe", "client\_ssn": "123-45-6789", "master\_rep\_id": "master123" } | Form controls client\_name, client\_ssn, and master\_rep\_id populated with API response values. |

**Test Purpose:**  
Ensure that the getAccountDetailsByAccountNo method handles API failures correctly by leaving the form controls empty and updating their enabled/disabled state appropriately.

**Test Description:**  
This unit test verifies that when the getAccountDetailsByAccountNo method is called and the API fails to fetch account details, the form controls (client\_name, client\_ssn, and master\_rep\_id) remain empty. Additionally, it ensures that client\_name stays enabled while client\_ssn and master\_rep\_id are disabled or hidden, maintaining a consistent UI state in the event of an API failure.

| **#** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.11 | Verify form controls remain empty and updated states on failure | API failure for account details retrieval | Mock API failure response: { "status": "failure" } | Form controls client\_name, client\_ssn, and master\_rep\_id remain empty. client\_name stays enabled, while client\_ssn and master\_rep\_id are disabled. |

**Test Purpose:**  
Ensure that the getAccountDetailsByAccountNo method correctly handles the case where no client details are returned, resetting the relevant form fields and clearing any temporary data.

**Test Description:**  
This test verifies that when the getAccountDetailsByAccountNo method is called and the API returns an empty list of client details (clients\_details\_data), the component correctly resets the account details and clears the relevant form fields. Additionally, it ensures that any temporary data stored in the component (such as tempData) is also cleared, maintaining a clean state.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.12 | Should reset account details when no client details are returned | API returns an empty list of client details | Mock API response: { "clients\_details\_data": [], "master\_repid": "" } | Form controls client\_name, client\_ssn, and master\_rep\_id are reset to empty values (""). accountDetails is reset to an empty array ([]). tempData is cleared to an empty array ([]). A console message "No clients found for the given account number!" is logged. |

**Test Purpose:**  
Ensure that the component correctly handles errors when the getAccountDetailsByAccountNo method encounters an issue, such as an API failure. It checks if the component calls the appropriate error handling methods (like resetting account details) and logs the error message to the console.

**Test Description:**  
This test ensures that the component correctly handles errors when the getAccountDetailsByAccountNo method encounters an issue, such as an API failure. It verifies if the component calls the appropriate error handling methods, such as resetting account details, and logs the error message ('Error Occured!') to the console.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.13 | Should handle error in getAccountDetailsByAccountNo | API failure simulation during the getAccountDetailsByAccountNo method call | Mock API rejection with error | The resetAccountDetails method is called to clear the component's data. The error message 'Error Occured!' along with the error details is logged to the console. All mocks are restored to their original state. |

**Test Purpose:**  
Ensure that when the getAccountDetailsByAccountNo method receives a response with a status other than 'success' (in this case, 'error'), the component correctly resets the account details and clears the form fields. This test also verifies that the resetAccountDetails method is called and the form controls are reset to empty values.

**Test Description:**  
This test ensures that when the getAccountDetailsByAccountNo method receives a response with a status other than 'success' (in this case, 'error'), the component will reset the account details and clear the form fields. It also verifies that the resetAccountDetails method is invoked and that the form controls (client\_name, client\_ssn, and master\_rep\_id) are reset to empty values.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.14 | Should reset account details and log when response status is not 'success' (error) | Response status is 'error', simulating an API failure | Mock API response with status: { "status": "error", "clients\_details\_data": [], "master\_repid": "" } | The resetAccountDetails method is called. The form controls (client\_name, client\_ssn, master\_rep\_id) are reset to empty values (""). The data is cleared, indicating the component responded to the error status. |

**Test Purpose:**  
Ensure that when a client is selected from the array list by their name, the corresponding SSN (client\_ssn) is correctly populated in the form. This test checks that the client\_ssn form control is updated with the correct value based on the selected client.

**Test Description:**  
This test verifies that when a client is selected from the array list by their name, the client\_ssn form control is updated with the correct SSN corresponding to the selected client. It ensures that the form field reflects the correct SSN for the chosen client.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.15 | Should set SSN for the selected client | Client is selected by name, SSN is correctly populated | Mock client data: { client\_name: 'John Doe', client\_ssn: '123-45-6789' } assigned to tempData | The client\_ssn form control is updated with '123-45-6789' when 'John Doe' is selected, confirming the correct SSN is set for the selected client. |

**Test Purpose:**  
Ensure that when the client name is changed (through a user action such as selecting from a dropdown or an input), the corresponding SSN (client\_ssn) is updated in the form. This test verifies that the client\_ssn field is correctly populated with the SSN value associated with the selected client name.

**Test Description:**  
This test ensures that when the client name changes (for example, by user selection), the client\_ssn form control is updated with the SSN of the selected client. It confirms that the form reflects the correct SSN after a client name change.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.16 | Should update the client SSN when client name changes | Client name changes, and SSN is updated accordingly | Mock event: { target: { value: 'John Doe' } } triggered for clientNameChange method, with tempData containing { client\_name: 'John Doe', client\_ssn: '123-45-6789' } | The client\_ssn form control is updated with '123-45-6789' when 'John Doe' is selected, confirming the correct SSN is populated after the name change. |

**Test Purpose:**  
Ensure that when the client name is changed to one that does not match the current client name in the form, the corresponding SSN (client\_ssn) is updated correctly. This test verifies that the client\_ssn form control is reset and updated to the SSN associated with the newly selected client.

**Test Description:**  
This test ensures that when the client name is changed to a name that does not match the current client name in the form, the client\_ssn form control is reset and updated to reflect the SSN of the newly selected client.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.17 | Should reset client SSN if client name does not match | Client name is changed, and SSN is updated accordingly | Mock event: { target: { value: 'Jane Doe' } } triggered for clientNameChange method, with tempData containing { client\_name: 'John Doe', client\_ssn: '123-45-6789' } and { client\_name: 'Jane Doe', client\_ssn: '987-65-4321' } | The client\_ssn form control is updated with '987-65-4321' when 'Jane Doe' is selected, ensuring the correct SSN is set based on the selected client name. |

**Test Purpose:**  
Ensure that if the client name entered or selected does not match any client in the available data (tempData), the client\_ssn field is reset. This test verifies that when a client name is changed to a non-existent client, the client\_ssn form control is cleared (reset to an empty string).

**Test Description:**  
This test ensures that if the client name is changed to one that does not exist in the available tempData, the client\_ssn field is reset to an empty string. This prevents displaying any SSN associated with a non-existent client.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.18 | Should reset client SSN if client name does not match any client | Client name is changed to one that does not exist in tempData, and SSN is reset | Mock event: { target: { value: 'Jane Doe' } } triggered for clientNameChange method, with tempData containing { client\_name: 'John Doe', client\_ssn: '123-45-6789' } | The client\_ssn form control is reset to an empty string (""), as 'Jane Doe' does not match any client in tempData. |

**Test Purpose:**  
Ensure that when the user does not have access to the account (i.e., no client details are returned from the API), certain form fields (like master\_rep\_id, client\_name, and client\_ssn) are hidden. This test verifies that the isVisible flag is set to false, indicating that these fields should not be displayed to the user.

**Test Description:**  
This test ensures that when the user does not have access to the account (and no client details are returned), the isVisible flag is set to false, effectively hiding the form fields (master\_rep\_id, client\_name, and client\_ssn) in the UI.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.19 | Should hide master rep id, client name, and client ssn when user does not have access to the account no | No client details available for the account, fields are hidden | Mock API response with { status: 'success', master\_repid: '123', clients\_details\_data: [] }, indicating no client details for the account | The isVisible flag is set to false, ensuring that the form fields (master\_rep\_id, client\_name, and client\_ssn) are hidden from the UI. |

**Test Purpose:**  
Ensure that when a truthy accountNo (e.g., '123456789') is provided, the component correctly processes it by setting the account\_no form control value, disabling it to prevent further modifications, and calling the necessary methods to fetch account details, reporter details, and transaction type. Additionally, it verifies that the isVisible flag is set to true, indicating that the relevant data is ready to be displayed.

**Test Description:**  
This test ensures that when a valid, truthy accountNo is provided, the component correctly processes it by setting the account\_no form control, disabling it, and calling methods to fetch data. Additionally, the test checks that the isVisible flag is set to true, confirming that the component is ready to display the relevant data.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.20 | Should handle truthy accountNo (If Branch) | Valid account number is provided, and relevant methods are called | Set accountNo to '123456789'. Spy on getAccountDetailsByAccountNo, getReporterDetails, and getTransactionType methods | The account\_no form control should be set to '123456789' and disabled. The methods getAccountDetailsByAccountNo, getReporterDetails, and getTransactionType should be called. The isVisible flag should be set to true, indicating that the data is ready to be displayed. |

**Test Purpose:**  
Ensure that when a falsy accountNo (e.g., an empty string) is provided, the component correctly handles the scenario by setting the isVisible flag to false (to hide the relevant data), enabling the account\_no form control (to allow further interaction), and calling the enable method on the form control.

**Test Description:**  
This test verifies that when a falsy accountNo is provided, the isVisible flag is set to false, the account\_no form control is enabled, and the enable method is called. This ensures that no data is displayed for a falsy account number, and the form control is available for further user input.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.21 | Should handle falsy accountNo (Else Branch) | Falsy accountNo is provided, relevant methods are called | Set accountNo to ''. Check that isVisible is set to false, enable method is called on account\_no form control | The isVisible flag should be false, ensuring the component does not display the relevant data. The account\_no form control should be enabled (i.e., disabled is false). The enable method should be called to allow further interaction with the form control. |

**Test Purpose:**  
Ensure that when the user does not have access to the account (i.e., the API returns a failure response), the component will display a pop-up and hide the account details. It verifies that the isVisible flag is set to false and the showDialog flag is set to true, indicating that the relevant pop-up should be displayed to inform the user they do not have access.

**Test Description:**  
This test ensures that if the user does not have access to the account (indicated by an API failure response), the component hides the account details by setting the isVisible flag to false and shows a pop-up (by setting the showDialog flag to true). This provides feedback to the user that they do not have access to the requested account.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.22 | Should show pop-up if user does not have access | API failure response indicating no access to account | Mock API response with status 'failure' to simulate lack of access | The isVisible flag should be false, confirming that the account details are hidden. The showDialog flag should be true, ensuring that a pop-up is displayed to inform the user they do not have access to the account. |

**Test Purpose:**  
Ensure that the applyPhoneNumberMask method is called correctly when the formatPhoneNumber method is executed. This test verifies that the phone number is formatted correctly (e.g., '123-456-7890') and that the applyPhoneNumberMask method receives the expected phone number value.

**Test Description:**  
This test ensures that when the formatPhoneNumber method is triggered, the applyPhoneNumberMask method is called with the correct phone number value. Additionally, it checks that the phone number is formatted as expected (e.g., '123-456-7890').

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.23 | Should call applyPhoneNumberMask correctly when formatting phone number | Phone number input event triggers the formatting process | Simulate a phone number input event with the value '1234567890'. Call formatPhoneNumber method | The applyPhoneNumberMask method should be called with the phone number '1234567890'. The event.target.value should be formatted as '123-456-7890', confirming that the phone number is correctly formatted. |

**Test Purpose:**  
Ensure that the formatPhoneNumberOnBlur method correctly formats the phone number when the input field loses focus. This test verifies that the applyPhoneNumberMask method is called with the correct phone number, and that the phone number is updated to the properly formatted value (e.g., '123-456-7890') using the setValue method.

**Test Description:**  
This test ensures that when the input field loses focus, the formatPhoneNumberOnBlur method is triggered. It checks if the applyPhoneNumberMask method is called with the correct phone number and verifies that the phone number is updated with the correctly formatted value.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.24 | Should handle formatPhoneNumberOnBlur correctly | Phone number is formatted when the input field loses focus | Initialize the reporter\_phone\_no form control with the value '1234567890'. Simulate the behavior of losing focus by calling formatPhoneNumberOnBlur. | The applyPhoneNumberMask method should be called with the phone number '1234567890'. The setValue method should be called with the formatted phone number '123-456-7890', confirming that the phone number is correctly formatted when the input field loses focus. |

**Test Purpose:**  
Ensure that the formatPhoneNumber method correctly formats a phone number containing non-numeric characters, such as parentheses and dashes. This test checks that the non-numeric characters are removed and the phone number is formatted correctly (e.g., '123-456-7890').

**Test Description:**  
This test ensures that when a phone number with non-numeric characters (like parentheses and dashes) is provided, the formatPhoneNumber method removes these characters and formats the phone number as '123-456-7890'.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.25 | Should format phone number with non-numeric characters | Phone number input containing non-numeric characters such as parentheses and dashes is formatted correctly. | Simulate an event with the phone number '(123) 456-7890'. Call the formatPhoneNumber method with the simulated event. | The phone number is correctly formatted as '123-456-7890', with non-numeric characters removed, confirming the phone number is formatted correctly. |

**Test Purpose:**  
Ensure that the formatPhoneNumber method does not alter the phone number if the input exceeds 10 digits. This test verifies that phone numbers longer than 10 digits are not formatted or changed.

**Test Description:**  
This test ensures that if a phone number input exceeds the standard 10 digits, the formatPhoneNumber method does not modify the phone number, leaving it unchanged.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.26 | Should not format phone number if input exceeds 10 digits | Phone number input exceeding 10 digits should not be formatted. | Simulate an event with the phone number '1234567890123'. Call the formatPhoneNumber method with the simulated event. | The event.target.value should remain '1234567890123', confirming that the phone number was not formatted and remained unchanged. |

**Test Purpose:**  
Ensure that the formatPhoneNumberOnBlur method skips formatting when the phone number field is empty. This test verifies that the setValue method is not called when the phone number field is empty.

**Test Description:**  
This test ensures that when the phone number field is empty, the formatPhoneNumberOnBlur method does not attempt to format the phone number. It confirms that no action is taken (i.e., setValue is not called) if the form control for the phone number is empty.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.27 | Should skip formatting on blur if phone number is empty | Ensure that the phone number is not formatted if the field is empty. | Initialize the reporter\_phone\_no form control with an empty string (''). Call the formatPhoneNumberOnBlur method. | The setValue method should not be called, confirming that the phone number is not formatted when the input field is empty. |

**Test Purpose:**  
Ensure that the formatPhoneNumberOnBlur method correctly formats the phone number when the input field loses focus (blur event).

**Test Description:**  
This test verifies that the formatPhoneNumberOnBlur method correctly formats the phone number and calls the applyPhoneNumberMask method with the expected phone number. It ensures that the phone number is formatted as '123-456-7890' upon the blur event.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.28 | Should format phone number correctly on blur event | Verify that the phone number is correctly formatted when the input field loses focus. | Set the value of the reporter\_phone\_no form control to '1234567890'. Call formatPhoneNumberOnBlur. | The applyPhoneNumberMask method should be called with the phone number '1234567890'. The phoneControl.value should be formatted as '123-456-7890'. |

**Test Purpose:**  
Verify that the getTransactionType method correctly populates the transaction types and subtypes in the component's state and sets the default value for the transaction\_type form control to '--Select Transaction Type--'.

**Test Description:**  
This test ensures that the getTransactionType method populates the transactionTypes array based on the response from the API. It also checks that the default value for the transaction\_type form control is set to '--Select Transaction Type--'.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.29 | Should populate transaction types and subtypes | Ensure that the getTransactionType method correctly populates transaction types and sets the default form control value. | Mock the getTransactionType API response with transaction types ('type1', 'type2') and subtypes ('subtype1', 'subtype2'). | The transactionTypes array should contain ['--Select Transaction Type--', 'type1', 'type2']. The transaction\_type form control should have the default value '--Select Transaction Type--'. |

**Test Purpose:**  
Verify that the subtypes are correctly reset when no transaction type is selected, and that the subtypes are populated when a valid transaction type is selected.

**Test Description:**  
This test ensures that when no transaction type is selected, the subtypes are reset to an empty state. It also checks that selecting a transaction type correctly populates the corresponding subtypes.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.30 | Should reset subtypes if the transaction type is not selected | Ensure that the subtypes are reset when no transaction type is selected. Verify that the subtypes are populated when a transaction type is selected. | Mock a successful response from the API with 'type1' and its subtypes ('subtype1', 'subtype2'). Simulate selecting 'type1' from the transaction\_type form control. | The transactionSubTypes array should contain ['--Select Transaction SubType--', 'subtype1', 'subtype2'] after selecting a transaction type. If no transaction type is selected, the subtypes array should be reset. |

**Test Purpose:**  
Ensure that when the transaction type is reset to the default value ('--Select Transaction Type--'), the corresponding subtypes are cleared.

**Test Description:**  
This test ensures that selecting the default option ('--Select Transaction Type--') for the transaction type resets the subtypes to an empty state.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.31 | Should reset subtypes when "Select Transaction Type" is selected | Verify that when the transaction type is reset to '--Select Transaction Type--', the subtypes are cleared. | Mock a successful API response with a transaction type ('type1') and its corresponding subtypes ('subtype1', 'subtype2'). Simulate selecting 'type1' and then resetting the selection to '--Select Transaction Type--'. | The transactionSubTypes array should be empty ([]), confirming that the subtypes have been cleared after resetting the transaction type. |

**Test** **Purpose**:

Ensure that when the transaction type is reset to the default value ('--Select Transaction Type--'), the corresponding subtypes are cleared.

**Test Description**:

This test ensures that selecting the default option ('--Select Transaction Type--') for the transaction type resets the subtypes to an empty state.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.32 | Should reset subtypes when "Select Transaction Type" is selected | Verify that when the transaction type is reset to '--Select Transaction Type--', the subtypes are cleared. | Mock a successful API response with a transaction type ('type1') and its corresponding subtypes ('subtype1', 'subtype2'). Simulate selecting 'type1' and then resetting the selection to '--Select Transaction Type--'. | The transactionSubTypes array should be empty ([]), confirming that the subtypes have been cleared after resetting the transaction type. |

**Test Purpose:**  
This test ensures that when the API call for fetching transaction types fails, the component resets the form values and logs the error.

**Test Description:**  
This test verifies that upon an API failure, the form values for transaction\_type, transaction\_sub\_type, and other\_suspicious\_activity\_details are reset to their default values. Additionally, the error is logged correctly to the console.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.33 | Should reset form values and log error when API call fails | Verify that when the API call fails, the form values are reset to their default values, and the error is logged. | Mock an error response from the getTransactionType API. Simulate the API failure by calling the getTransactionType method. | The form values for transaction\_type, transaction\_sub\_type, and other\_suspicious\_activity\_details should be reset to their default values: ['--Select Transaction Type--'], ['--Select Transaction SubType--'], and an empty string, respectively. The error message should be logged to the console using console.log. |

**Test Purpose:**  
This test ensures that when the API call for fetching transaction types succeeds, the default values for transaction\_type and transaction\_sub\_type are set correctly in the form.

**Test Description:**  
This test verifies that when the API response is successful, the transaction\_type and transaction\_sub\_type form controls are initialized with the correct default values.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.34 | Should set default value for transaction type and sub type when API is successful | Verify that the default values for transaction\_type and transaction\_sub\_type are set correctly when the API call is successful. | Mock a successful API response with transaction types ('type1') and subtypes ('subtype1', 'subtype2'). Call the getTransactionType method. | The transaction\_type form control should have the default value '--Select Transaction Type--'. The transaction\_sub\_type form control should have the default value '', indicating no subtype is selected initially. |

**Test Purpose:**  
This test verifies that the showSeniorInvestorInvolvedFlag is set to false when the selectedTransactionType is initially empty. This ensures the flag starts with the correct value when no transaction type is selected.

**Test Description:**  
This test ensures that the showSeniorInvestorInvolvedFlag is set to false if no transaction type is selected initially.

| **S.No.** | **Test Case For** | | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- | --- |
| 1.35 | | Should set showSeniorInvestorInvolvedFlag to false initially | Verify that the showSeniorInvestorInvolvedFlag is false when no transaction type is selected. | Set the selectedTransactionType to an empty string. Call the transactionTypeChange method. | The showSeniorInvestorInvolvedFlag should be false. |

**Test Purpose:**  
This test checks that the showSeniorInvestorInvolvedFlag is set to true when the selectedTransactionType is 'Electronic Transfer'. This verifies that the flag is correctly updated for this specific transaction type.

**Test Description:**  
This test ensures that the showSeniorInvestorInvolvedFlag is set to true if the selectedTransactionType is 'Electronic Transfer'.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.36 | Should set showSeniorInvestorInvolvedFlag to true if selectedTransactionType is "Electronic Transfer" | Verify that the showSeniorInvestorInvolvedFlag is set to true when the selectedTransactionType is 'Electronic Transfer'. | Set the selectedTransactionType to 'Electronic Transfer'. Call the transactionTypeChange method. | The showSeniorInvestorInvolvedFlag should be true. |

**Test Purpose:**  
This test ensures that the showSeniorInvestorInvolvedFlag is set to true when the selectedTransactionType is 'Check Fraud'. It verifies that the flag responds correctly to this transaction type.

**Test Description:**  
This test checks that the showSeniorInvestorInvolvedFlag is set to true if the selectedTransactionType is 'Check Fraud'.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.37 | Should set showSeniorInvestorInvolvedFlag to true if selectedTransactionType is "Check Fraud" | Verify that the showSeniorInvestorInvolvedFlag is set to true when the selectedTransactionType is 'Check Fraud'. | Set the selectedTransactionType to 'Check Fraud'. Call the transactionTypeChange method. | The showSeniorInvestorInvolvedFlag should be true. |

**Test Purpose:**  
This test ensures that the showSeniorInvestorInvolvedFlag remains false when the selectedTransactionType is neither 'Electronic Transfer' nor 'Check Fraud'. It verifies that the flag does not erroneously get set to true for other transaction types.

**Test Description:**  
This test checks that the showSeniorInvestorInvolvedFlag is not set to true when the selectedTransactionType is a different type, such as 'Identity Theft', which is neither 'Electronic Transfer' nor 'Check Fraud'.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.38 | Should keep showSeniorInvestorInvolvedFlag false if selectedTransactionType is neither "Electronic Transfer" nor "Check Fraud" | Verify that the showSeniorInvestorInvolvedFlag remains false when the selectedTransactionType is not 'Electronic Transfer' or 'Check Fraud'. | Set the selectedTransactionType to 'Identity Theft'. Call the transactionTypeChange method. | The showSeniorInvestorInvolvedFlag should remain false. |

**Test Purpose:**  
This test verifies that the showSeniorInvestorInvolvedFlag is set to true when the selectedTransactionType is 'Identity Theft' and the transactionSubType is 'Advisor or Client Impersonation'.

**Test Description:**  
This test ensures that the showSeniorInvestorInvolvedFlag is correctly updated to true when the selected transaction type is 'Identity Theft' and the sub-type is 'Advisor or Client Impersonation'.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.39 | Should set showSeniorInvestorInvolvedFlag to true for "Identity Theft" with "Advisor or Client Impersonation" | Verify that the showSeniorInvestorInvolvedFlag is set to true when selectedTransactionType is 'Identity Theft' and transactionSubType is 'Advisor or Client Impersonation'. | Set selectedTransactionType to 'Identity Theft'. Simulate the change event for transactionSubType with the value 'Advisor or Client Impersonation'. | The showSeniorInvestorInvolvedFlag should be true. |

**Test Purpose:**  
This test ensures that the showSeniorInvestorInvolvedFlag is set to true when the selectedTransactionType is 'Other Suspicious Activity' and the transactionSubType is 'Elder Financial Exploitation'.

**Test Description:**  
This test verifies that the showSeniorInvestorInvolvedFlag is correctly set to true when the selected transaction type is 'Other Suspicious Activity' and the sub-type is 'Elder Financial Exploitation'.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.40 | Should set showSeniorInvestorInvolvedFlag to true for "Other Suspicious Activity" with "Elder Financial Exploitation" | Verify that the showSeniorInvestorInvolvedFlag is set to true when selectedTransactionType is 'Other Suspicious Activity' and transactionSubType is 'Elder Financial Exploitation'. | Set selectedTransactionType to 'Other Suspicious Activity'. Simulate the change event for transactionSubType with the value 'Elder Financial Exploitation'. | The showSeniorInvestorInvolvedFlag should be true. |

**Test Purpose:**  
This test ensures that the showSeniorInvestorInvolvedFlag is set to true when the selectedTransactionType is 'Other Suspicious Activity' and the transactionSubType is 'Other'.

**Test Description:**  
This test verifies that the showSeniorInvestorInvolvedFlag is correctly set to true when the selectedTransactionType is 'Other Suspicious Activity' and the transactionSubType is 'Other'.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.41 | Should set showSeniorInvestorInvolvedFlag to true for "Other Suspicious Activity" with "Other" | Verify that the showSeniorInvestorInvolvedFlag is set to true when the selectedTransactionType is 'Other Suspicious Activity' and the transactionSubType is 'Other'. | Set selectedTransactionType to 'Other Suspicious Activity'. Simulate the change event for transactionSubType with the value 'Other'. | The showSeniorInvestorInvolvedFlag should be true. |

**Test Purpose:**  
This test ensures that the showSeniorInvestorInvolvedFlag remains false when none of the conditions for setting it to true are met.

**Test Description:**  
This test verifies that the showSeniorInvestorInvolvedFlag is set to false when the selectedTransactionType and transactionSubType do not match any predefined values. It ensures that the flag is not erroneously set to true in such cases.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.42 | Should set showSeniorInvestorInvolvedFlag to false when no conditions match | Verify that the showSeniorInvestorInvolvedFlag remains false when no matching conditions for selectedTransactionType or transactionSubType are found. | Set selectedTransactionType to 'Non-matching Type'. Simulate the change event for transactionSubType with a random value. | The showSeniorInvestorInvolvedFlag should remain false. |

**Test Purpose:**  
This test ensures that the accountDetails and clientNames are reset to empty arrays when the account number is cleared.

**Test Description:**  
This test verifies that when the account number is cleared, the accountDetails and clientNames are reset to empty arrays, ensuring the correct behavior when there is no account number.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.43 | Should reset account details when account number is empty | Verify that when the account number is cleared, the accountDetails and clientNames are reset to empty arrays. | Set the account\_no field in the form to an empty string. Call the getAccountDetailsByAccountNo() method. | The accountDetails and clientNames should both be empty arrays. |

**Test Purpose:**  
This test ensures that the component handles errors from the API correctly, specifically verifying that accountDetails remains empty when the API call fails.

**Test Description:**  
This test verifies that when the API call to fetch account details fails, the accountDetails array is reset to an empty array, confirming that the component correctly handles API errors.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.44 | Should handle API errors correctly | Verify that the accountDetails array remains empty when the API call fails. | Mock the getAccountDetails API method to reject with an error. Call the getAccountDetailsByAccountNo() method. | The accountDetails should remain an empty array after the error is thrown. |

**Test Purpose:**  
This test ensures that the referral form is submitted correctly and reset after a successful submission. It verifies that after submission, the form fields are cleared, indicating successful reset.

**Test Description:**  
This test verifies that when the referral form is submitted successfully, the form values are sent to the API and reset to their default state. It checks that the form is cleared after a successful submission.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.45 | Should submit the referral form and reset on success | Verify that the form is submitted and reset after successful submission. | Set mock data for the referralForm and enable the form controls for submission. Mock successful API responses for necessary API calls (submit form, get reporter details, transaction types, and account details). Call submitReferralForm(). | The form data should be submitted to the API. After submission, the form values should reset to their default state. |

**Test Purpose:**  
This test ensures that when an error response is received from the API, the form is reset and an appropriate error message is logged.

**Test Description:**  
This test verifies that if an error response is encountered, the referral form is reset to its initial state and an error message is logged using console.log().

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.46 | Should handle error response and log message | Verify that the form is reset and an error message is logged when an error response is received. | Set up a mock error response with status 'error'. Check if resetReferralForm() is called when the error is received. | resetReferralForm() should be called, and the error message should be logged using console.log(). |

**Test Purpose:**  
This test ensures that the form is not submitted if it is invalid. It verifies that the form's invalid state is respected during the submission process.

**Test Description:**  
This test checks that if the form has invalid data (e.g., missing required fields), it should not be submitted. It ensures that no submission happens if the form is not filled out correctly.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.47 | Should not submit form if invalid | Verify that the form is not submitted if it is invalid. | Set the form values with invalid data (missing required fields). Call submitReferralForm(). | The form should remain invalid, and no form submission should occur if the form is not filled out correctly. |

**Test Purpose:**  
This test ensures that the value of the trades\_placed\_flag form field is correctly set when the form is submitted.

**Test Description:**  
This test verifies that after the form is submitted, the trades\_placed\_flag field retains the correct value ('Y').

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.48 | Should set the correct value for trades\_placed\_flag | Verify that the trades\_placed\_flag form field retains the value 'Y' after form submission. | Set the value of the trades\_placed\_flag field to 'Y' and call submitReferralForm(). | The form's trades\_placed\_flag field should hold the value 'Y' after submission. |

**Test Purpose:**  
This test ensures that the form is correctly reset after calling resetReferralForm().

**Test Description:**  
This test verifies that when resetReferralForm() is called, the form values are reset to their default state, including resetting the account\_no field to an empty string and the submitted flag to false.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.49 | Should reset the form correctly | Verify that after calling resetReferralForm(), the form is reset, and the account\_no field and submitted flag are correctly reset. | Call the resetReferralForm() method. | The account\_no field should be reset to an empty string, and the submitted flag should be set to false. |

**Test Purpose:**  
This test ensures that invalid form controls are logged and that a general invalidity message is logged when the form is invalid.

**Test Description:**  
This test verifies that when the form is invalid, the invalid form controls are logged, and a general invalidity message is logged. The goal is to confirm that the form’s invalid state is handled correctly with appropriate console logs.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.50 | Should log invalid controls if form is invalid | Verify that invalid form controls are logged and a general invalidity message is shown when the form is invalid. | Set the form values to invalid (e.g., empty or null fields). Spy on console.log() to capture the output. Call submitReferralForm(). | console.log() should be called with a list of invalid form controls and a message like 'Form is invalid. Please correct the errors and try again.' |

**Test Purpose:**  
This test ensures that the component handles an API failure correctly, logging the error message when the API request fails.

**Test Description:**  
This test verifies that when the API fails, the error message is logged in the console. It confirms that the error handling mechanism is triggered correctly and the expected error message is displayed.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.51 | Should handle API failure and show error message | Verify that the error message is logged when the API request fails. | Set the form to a valid state. Mock the submitReferralForm() method to reject with an error. Spy on console.log() to capture the error message. Call submitReferralForm(). | The error message 'Error Occurred!' should be logged to the console. |

**Test Purpose:**  
This test ensures that certain form controls are enabled before the form is submitted, verifying that the correct controls are enabled during the submission process.

**Test Description:**  
This test checks that specific form controls are enabled before submission. It ensures that the form controls, like master\_rep\_id and client\_ssn, are properly enabled before the form is submitted, confirming that the form setup is correct for submission.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.52 | Should enable form controls before submission | Verify that the form controls are enabled before submission. | Set the form to valid values. Mock the enable() method for relevant form controls. Call submitReferralForm(). | The enable() method should be called on the relevant form controls (e.g., master\_rep\_id, client\_ssn, etc.) before submission. |

**Test Purpose:**  
This test verifies that pressing keys other than the ones meant to trigger the btnReset method (like the "A" key) does not trigger the reset functionality.

**Test Description:**  
This test ensures that the btnReset method is only triggered by specific keys (e.g., the "Enter" key) and not by other keys like the "A" key. It verifies that when a non-reset key is pressed, the reset functionality is not activated, and the default behavior of the key press is maintained.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.53 | onKeyDown - Should not call btnReset on other key presses | Verify that non-reset keys do not trigger the btnReset method. | Create a keyboard event for the "A" key (keyCode 65). Spy on the btnReset method and preventDefault() method. Call onKeyDown() with the keyboard event. | The btnReset method should not be called, and the preventDefault() method should not be triggered when a non-reset key (e.g., "A") is pressed. |

**Test Purpose:**  
This test ensures that the btnReset method in the component correctly navigates to the root path and resets both the account details and the referral form.

**Test Description:**  
This test verifies that when the btnReset method is invoked, it navigates to the root path, resets the account details, and clears the referral form by calling the respective reset methods.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.54 | btnReset - Should navigate to root path and call reset methods | Verify that the btnReset method navigates to the root path and resets the account details and referral form. | Spy on the router.navigate, resetAccountDetails, and resetReferralForm methods. Call btnReset() to simulate the reset action. | The router.navigate should be called with [''] to navigate to the root path. The resetAccountDetails method should be called to clear account details, and the resetReferralForm method should be invoked to reset the referral form. |

**Test Purpose:**  
This test ensures that the btnReset method correctly calls the router.navigate method to navigate to the root path and invokes both the resetAccountDetails and resetReferralForm methods to reset the account details and referral form.

**Test Description:**  
This test verifies that the btnReset method, when called, navigates to the root path, resets the account details, and clears the referral form by invoking the appropriate reset methods.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.55 | btnReset - Should navigate to root path and reset form and account details | Verify that the btnReset method calls router.navigate, resetAccountDetails, and resetReferralForm methods. | Spy on the router.navigate, resetAccountDetails, and resetReferralForm methods. Call btnReset() to simulate the reset action. | The router.navigate should be called with the argument [''] to navigate to the root path. The resetAccountDetails method should be invoked to reset the account details. The resetReferralForm method should be invoked to reset the referral form. |

**Test Purpose:**  
This test ensures that the btnReset method behaves as expected by navigating to the root path and calling the resetAccountDetails and resetReferralForm methods to reset the form and account details.

**Test Description:**  
This test verifies that when the btnReset method is invoked, it correctly navigates to the root path and resets the account details and the referral form by calling the respective methods.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.56 | btnReset - Should navigate to root path and reset form and account details | Verify that the btnReset method calls router.navigate and invokes both resetAccountDetails and resetReferralForm methods. | Mock the resetAccountDetails and resetReferralForm methods using jest.fn(). Call btnReset() to simulate the reset action. | The router.navigate should be called with the argument [''] to navigate to the root path. The resetAccountDetails method should be called to reset account details. The resetReferralForm method should be called to reset the referral form. |

**Test Purpose:**  
This test ensures that the resetAccountDetails method correctly resets the account details and form values to their initial state.

**Test Description:**  
This test verifies that when the resetAccountDetails method is invoked, the component's account details, form values, and relevant form controls are reset to their default values.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.57 | resetAccountDetails - Should reset account details and form values correctly | Verify that calling the resetAccountDetails method resets the account details and form values to their initial state. | Call the resetAccountDetails() method. | The accountDetails property should be reset to an empty array ([]). The clientNames property should be reset to an empty array ([]). The selectedClientName property should be reset to an empty string (''). The form controls client\_name, client\_ssn, and master\_rep\_id should be cleared and set to empty strings (''). |

**Test Purpose:**  
This test ensures that the resetReferralForm method correctly resets the referral form values to their initial state and calls the necessary API methods, such as getReporterDetails and getTransactionType.

**Test Description:**  
This test verifies that when the resetReferralForm method is invoked, the referral form's values are reset to their default values, and necessary API methods are triggered as expected.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.58 | resetReferralForm - Should reset referral form and call necessary API methods | Verify that calling the resetReferralForm method resets the form values and calls the necessary API methods. | Call the resetReferralForm() method. | All form controls (e.g., account\_no, master\_rep\_id, client\_name, client\_ssn, reporter\_first\_name, reporter\_last\_name, reporter\_email, reporter\_phone\_no, transaction\_type, transaction\_sub\_type, trades\_placed\_flag, other\_suspicious\_activity\_details, additional\_transaction\_details) should be reset to their initial values (empty strings or null as applicable). The submitted flag should be set to false. The showSeniorInvestorInvolvedFlag should be set to false. The isVisible flag should be set to false. The getReporterDetails and getTransactionType methods should be called. |

**Test Purpose:**  
This test ensures that the onConfirm method correctly sets the showDialog property to false, indicating that the dialog is closed.

**Test Description:**  
This test verifies that when the onConfirm method is called, the showDialog property is updated to false, indicating that the dialog box has been closed.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.59 | onConfirm - Should set showDialog to false | Verify that the onConfirm method correctly sets the showDialog property to false. | Set the showDialog property to true and call the onConfirm() method. | The showDialog property should be set to false after calling the onConfirm method. |

**Test Purpose:**  
This test ensures that the onCloseDialog method correctly sets the showReferenceNumber property to false, indicating that the reference number dialog has been closed.

**Test Description:**  
This test verifies that when the onCloseDialog method is called, the showReferenceNumber property is updated to false, closing the reference number dialog.

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.60 | onCloseDialog - Should set showReferenceNumber to false | Verify that the onCloseDialog method correctly sets the showReferenceNumber property to false. | Set the showReferenceNumber property to true and call the onCloseDialog() method. | The showReferenceNumber property should be set to false after calling the onCloseDialog method. |

**Test Purpose:**

This test ensures that the client\_name field’s validation is updated dynamically based on the user’s access to the account number. When the user has access, the field should be required, and when the user does not have access, the field should not be required.

**Test Description:**

This test verifies that when the user has access to the account number, the client\_name field becomes required. When the user does not have access, the client\_name field should not be required.

**Test Cases**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.61 | toggleClientNameValidation - Should Make client\_name Required When User Has Access | Verify that the client\_name field becomes required when the user has access to the account number. | Call the toggleClientNameValidation(true) method to simulate access. Retrieve the client\_name form control from the referral form. | The client\_name field should have the Validators.required validator applied, making the field required. |

**Test Purpose:**

This test ensures that the client\_name field's validation is updated dynamically based on the user’s access to the account number. When the user has access, the field should be required, and when the user does not have access, the field should not be required.

**Test Description:**

This test verifies that when the user has access to the account number, the client\_name field becomes required. When the user does not have access, the client\_name field should not be required.

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.62 | toggleClientNameValidation - Should Make client\_name Required When User Has Access | Verify that the client\_name field becomes required when the user has access to the account number. | Call the toggleClientNameValidation(true) method to simulate access. Retrieve the client\_name form control from the referral form. | The client\_name field should have the Validators.required validator applied, making the field required. |
|  | toggleClientNameValidation - Should Make client\_name Optional When User Does Not Have Access | Verify that the client\_name field becomes optional when the user does not have access to the account number. | Call the toggleClientNameValidation(false) method to simulate no access. Retrieve the client\_name form control from the referral form. | The client\_name field should not have the Validators.required validator applied, making the field optional. |

**Test Purpose:**

This test ensures that the client\_name field's validation is updated dynamically based on the isVisible flag. Specifically, it ensures that when isVisible is true, the field becomes required, and when isVisible is false, the field is not required.

**Test Description:**

This test verifies that the client\_name field's validation is updated dynamically based on the isVisible flag. It ensures that when isVisible is true, the field becomes required, and when isVisible is false, the field is not required.

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 1.63 | toggleClientNameValidation - Should Update Validation When isVisible is False | Verify that the client\_name field is not required when isVisible is false. | Set isVisible to false, then call toggleClientNameValidation(false). Set the client\_name field value to an empty string (''). | The client\_name field should be valid even with an empty value, as it is not required when isVisible is false. |
|  | toggleClientNameValidation - Should Update Validation When isVisible is True | Verify that the client\_name field is required when isVisible is true. | Set isVisible to true, then call toggleClientNameValidation(true). Set the client\_name field value to an empty string (''). | The client\_name field should be invalid because it is required when isVisible is true and the field is empty. |
|  | toggleClientNameValidation - Should Be Valid When client\_name Has a Valid Value | Verify that the client\_name field becomes valid when a valid value is entered. | Set a valid value (e.g., 'John Doe') for the client\_name field after calling toggleClientNameValidation(true). | The client\_name field should be valid when it has a valid value (e.g., 'John Doe'), regardless of isVisible. |

**API Unit Test**

**Test Purpose:**

This test ensures that the getAccountDetails method in the service correctly calls Rest.get with the proper URL and returns the expected transformed account details.

**Test Description:**

This test verifies that the getAccountDetails method correctly calls the Rest.get API with the appropriate URL, including the account number. It also ensures that the method returns the transformed account details, which exclude unnecessary data and only return relevant information like master\_repid and clients\_details\_data.

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 2.1 | getAccountDetails - Should Call Rest.get with Correct URL | Verify that Rest.get is called with the correct URL constructed using the accountNumber. | Define a mock accountNumber ('65497603'). Mock the Rest.get method to return a simulated response. Call getAccountDetails(accountNumber) with the mock data. | The Rest.get method should be called with the correct URL: ${environment.ReferralAPI}account-details/65497603. |
|  | getAccountDetails - Should Return Transformed Account Details | Verify that the returned account details are correctly transformed and exclude unnecessary data. | Define a mock response simulating account details and client information, including master\_repid and clients\_details\_data. Call getAccountDetails(accountNumber). | The returned result should include master\_repid and clients\_details\_data in the correct format, excluding unnecessary data. |

**Test Purpose:**

This test ensures that the getReporterDetails method in the service correctly calls Rest.get with the appropriate URL and returns the transformed reporter details.

**Test Description:**

This test verifies that the getReporterDetails method calls Rest.get with the correct URL and transforms the response to return the reporter's first name, last name, and email address in the correct format.

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 2.2 | getReporterDetails - Should Call Rest.get with Correct URL | Verify that Rest.get is called with the correct URL: ${environment.ReferralAPI}reporter-details. | Define a mock response for the API, simulating the data structure returned when calling the getReporterDetails API. Call service.getReporterDetails(). | The Rest.get method should be called with the URL: ${environment.ReferralAPI}reporter-details. |
|  | getReporterDetails - Should Return Transformed Reporter Details | Verify that the returned reporter details are correctly mapped. | Define a mock response simulating reporter details with first\_name, last\_name, and email. Call service.getReporterDetails(). | The returned result should correctly map the fields as follows: reporter\_first\_name → 'John', reporter\_last\_name → 'Doe', reporter\_email → 'john.doe@lplfinancial.com'. |

**Test Purpose:**

This test ensures that the getTransactionType method in the service correctly calls Rest.get with the appropriate URL and returns the transformed transaction types data.

**Test Description:**

This test verifies that the getTransactionType method correctly calls Rest.get with the correct URL and transforms the API response to return the transaction types in the expected structure.

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 2.3 | getTransactionType - Should Call Rest.get with Correct URL | Verify that Rest.get is called with the correct URL: ${environment.ReferralAPI}transaction-types. | Define a mock response that simulates the structure of the API response, including transaction\_types\_info with categories and subtypes. Call service.getTransactionType(). | The Rest.get method should be called with the URL: ${environment.ReferralAPI}transaction-types. |
|  | getTransactionType - Should Return Transformed Transaction Types | Verify that the returned transaction types match the expected structure. | Define a mock response with transaction\_types\_info mapping categories (e.g., "Electronic Transfer", "Check Fraud") to transaction subtypes. Call service.getTransactionType(). | The result should return a transactionType object containing a transaction\_types\_info object, mapping categories to subtypes, e.g., "Electronic Transfer" → ["ACH Transaction", "Wire Transaction"]. |

**Test Purpose:**

This test ensures that the submitReferralForm method calls Rest.post with the correct URL and successfully processes the response, returning the expected case details status and reference number.

**Test Description:**

This test verifies that the submitReferralForm method correctly calls Rest.post with the appropriate URL and form data, and returns the expected response containing the status ("success") and reference number (e.g., 126).

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 2.4 | submitReferralForm - Should Call Rest.post with Correct URL | Verify that Rest.post is called with the correct URL and form data. | Create an instance of ICaseDetailsModelRequest with valid form data. Use jest.mockResolvedValue to mock a successful API response. Call service.submitReferralForm(formData). | The Rest.post method should be called with the URL: ${environment.ReferralAPI}case-details and the correct form data. |
|  | submitReferralForm - Should Return Correct Status and Reference Number | Verify that the response contains the correct status and reference number. | Mock the API response to return a status of "success" and a reference\_number (e.g., 126). Call service.submitReferralForm(formData). | The result should return a status of "success" and a reference\_number of 126. |

**Test Purpose:**

This test ensures that the submitReferralForm method handles cases where the response structure is invalid or missing expected data.

**Test Description:**

This test verifies that the submitReferralForm method correctly handles invalid responses, such as missing data or missing expected fields, by returning default values (status: '', reference\_number: '').

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 2.5 | submitReferralForm - Should Handle Invalid Response Structure | Verify that the method handles an invalid response structure (e.g., missing data) | Create an instance of ICaseDetailsModelRequest with valid form data. Mock an invalid response (e.g., missing data or data.data). Call service.submitReferralForm(formData). | The Rest.post method should be called with the correct URL and form data. The method should return default values: status: '' and reference\_number: ''. |
|  | submitReferralForm - Should Return Default Values for Invalid Response | Verify that default values are returned when the response structure is invalid. | Mock an invalid response structure (e.g., missing the data or data.data fields). Call service.submitReferralForm(formData). | The result should return default values: status: '' and reference\_number: '' when the response is missing expected data. |

This test ensures that the submitReferralForm method gracefully handles cases where the API response is empty (null or undefined).

**Test Description:**

This test verifies that the submitReferralForm method correctly handles empty responses, such as when the response is null or undefined, and returns default values (status: '', reference\_number: '').

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 2.6 | submitReferralForm - Should Handle Empty Response | Verify that the method handles an empty response (null or undefined). | Create an instance of ICaseDetailsModelRequest with valid form data. Mock an empty response (null) using mockRestPost.mockResolvedValue(null). Call service.submitReferralForm(formData). | The method should return default values: status: '' and reference\_number: '' when the response is null or undefined. |

**Test Purpose:**

This test ensures that the submitReferralForm method handles error responses, such as network errors, gracefully.

**Test Description:**

This test verifies that the submitReferralForm method correctly handles error responses, such as network errors, by catching the error and ensuring that no unhandled exceptions occur. The error should be an instance of Error with the appropriate message.

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 2.7 | submitReferralForm - Should Handle Error Response Gracefully | Verify that the method handles error responses (e.g., network errors) gracefully. | Create an instance of ICaseDetailsModelRequest with valid form data. Mock a rejected Rest.post response with an error (e.g., new Error('Network error')). Call service.submitReferralForm(formData). | The error should be handled gracefully, and the error should be an instance of Error with the message "Network error". No unhandled exceptions should occur. |

**Validator Unit Test**

**Test Purpose:**

This test ensures that the lplAccountValidator function returns null when a valid account number is provided. A valid account number is alphanumeric and has 10 or fewer characters.

**Test Description:**

This test verifies that the lplAccountValidator function returns null for valid account numbers, indicating that there are no validation errors.

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 3.1 | lplAccountValidator - Should Return Null for a Valid Account | Verify that the lplAccountValidator returns null for a valid account number. | Create a mock control with a valid alphanumeric account number ('Valid123'). Call ReferralCustomValidators.lplAccountValidator(control) with the mock control. | The validator should return null, indicating that the account number is valid and there are no errors. |

**Test Purpose:**

This test ensures that the lplAccountValidator function returns an error object when the account number exceeds 10 characters.

**Test Description:**

This test verifies that the lplAccountValidator function returns an error object { invalidAccount: true } for account numbers that are longer than 10 characters.

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 3.2 | lplAccountValidator - Should Return Error Object for an Invalid Account (More Than 10 Characters) | Verify that the lplAccountValidator returns an error object for account numbers longer than 10 characters. | Create a mock control with an invalid account number ('InvalidAccount123'). Call ReferralCustomValidators.lplAccountValidator(control) with the mock control. | The validator should return the error object { invalidAccount: true } for account numbers longer than 10 characters. |

**Test Purpose:**

This test ensures that the lplAccountValidator function returns an error object when the account number contains non-alphanumeric characters.

**Test Description:**

This test verifies that the lplAccountValidator function returns an error object { invalidAccount: true } for account numbers that contain non-alphanumeric characters.

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 3.3 | lplAccountValidator - Should Return Error Object for an Invalid Account (Non-Alphanumeric Characters) | Verify that the lplAccountValidator returns an error object for account numbers containing non-alphanumeric characters. | Create a mock control with an account number containing special characters ('Invalid@123'). Call ReferralCustomValidators.lplAccountValidator(control) with the mock control. | The validator should return the error object { invalidAccount: true } for account numbers containing non-alphanumeric characters. |

**Test Purpose:**

This test ensures that the lplAccountValidator function returns an error object when the account number is empty.

**Test Description:**

This test verifies that the lplAccountValidator function returns an error object { invalidAccount: true } for empty account numbers.

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 3.4 | lplAccountValidator - Should Return Error Object for an Empty Account | Verify that the lplAccountValidator returns an error object for an empty account number. | Create a mock control with an empty account number (''). Call ReferralCustomValidators.lplAccountValidator(control) with the mock control. | The validator should return the error object { invalidAccount: true } when the account number is empty. |

**Test Purpose:**

This test ensures that the phoneValidator function returns null when a valid phone number is provided. A valid phone number should be in the format xxx-xxx-xxxx where x represents a digit.

**Test Description:**

This test verifies that the phoneValidator function correctly validates a phone number in the required format and returns null for valid phone numbers, indicating no errors.

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 3.5 | phoneValidator - Should Return Null for a Valid Phone Number | Verify that the phoneValidator returns null for a valid phone number. | Create a mock control with a valid phone number ('123-456-7890'). Call ReferralCustomValidators.phoneValidator(control) with the mock control. | The validator should return null, indicating that the phone number is valid and there are no errors. |

**Test Purpose:**

This test ensures that the phoneValidator function returns an error object when the phone number has too few digits (less than 10 digits).

**Test Description:**

This test verifies that the phoneValidator function returns an error object { invalidPhone: true } for phone numbers that have fewer than 10 digits.

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 3.6 | phoneValidator - Should Return Error Object for an Invalid Phone Number (Too Few Digits) | Verify that the phoneValidator returns an error object for phone numbers with too few digits. | Create a mock control with an invalid phone number ('123-45-789'). Call ReferralCustomValidators.phoneValidator(control) with the mock control. | The validator should return the error object { invalidPhone: true } for phone numbers with fewer than 10 digits. |

**Test Purpose:**

This test ensures that the phoneValidator function returns an error object when the phone number is in the wrong format (e.g., using slashes instead of dashes).

**Test Description:**

This test verifies that the phoneValidator function returns an error object { invalidPhone: true } when the phone number is in an invalid format, such as using slashes instead of dashes.

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 3.7 | phoneValidator - Should Return Error Object for an Invalid Phone Number (Wrong Format) | Verify that the phoneValidator returns an error object for phone numbers with an invalid format. | Create a mock control with an invalid phone number ('123/456/7890'). Call ReferralCustomValidators.phoneValidator(control) with the mock control. | The validator should return the error object { invalidPhone: true } for phone numbers with an invalid format (e.g., slashes). |

**Test Purpose:**

This test ensures that the phoneValidator function returns an error object when the phone number is empty.

**Test Description:**

This test verifies that the phoneValidator function returns an error object { invalidPhone: true } when the phone number is empty.

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 3.8 | phoneValidator - Should Return Error Object for an Empty Phone Number | Verify that the phoneValidator returns an error object for an empty phone number. | Create a mock control with an empty phone number (''). Call ReferralCustomValidators.phoneValidator(control) with the mock control. | The validator should return the error object { invalidPhone: true } when the phone number is empty. |

**Test Purpose:**

This test ensures that the emailValidator function returns null when the provided email address is valid and ends with @lplfinancial.com.

**Test Description:**

This test verifies that the emailValidator function correctly identifies a valid email address that ends with @lplfinancial.com and returns null, indicating no validation errors.

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 3.9 | emailValidator - Should Return Null for a Valid Email Address with @lplfinancial.com | Verify that the emailValidator returns null for a valid email address ending with @lplfinancial.com. | Create a mock control with a valid email address ('valid@lplfinancial.com'). Call ReferralCustomValidators.emailValidator(control) with the mock control. | The validator should return null, indicating the email address is valid. |

**Test Purpose:**

This test ensures that the emailValidator function returns null when the provided email address is valid and ends with @lpl.com.

**Test Description:**

This test verifies that the emailValidator function correctly identifies a valid email address that ends with @lpl.com and returns null, indicating no validation errors.

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 3.10 | emailValidator - Should Return Null for a Valid Email Address with @lpl.com | Verify that the emailValidator returns null for a valid email address ending with @lpl.com. | Create a mock control with a valid email address ('valid@lpl.com'). Call ReferralCustomValidators.emailValidator(control) with the mock control. | The validator should return null, indicating the email address is valid. |

**Test Purpose:**

This test ensures that the emailValidator function returns an error object when the provided email address is empty.

**Test Description:**

This test verifies that the emailValidator function correctly identifies an empty email address and returns an error object { invalidEmail: true }.

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 3.11 | emailValidator - Should Return Error Object for an Empty Email Address | Verify that the emailValidator returns an error object { invalidEmail: true } for an empty email address. | Create a mock control with an empty email address (''). Call ReferralCustomValidators.emailValidator(control) with the mock control. | The validator should return the error object { invalidEmail: true } when the email address is empty. |

**Test Purpose:**

This test ensures that the emailValidator function returns an error object when the provided email address has a missing domain part.

**Test Description:**

This test verifies that the emailValidator function correctly identifies an email address with a missing domain part (e.g., invalid@.com) and returns an error object { invalidEmail: true }.

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 3.12 | emailValidator - Should Return Error Object for an Email Address with a Missing Domain Part | Verify that the emailValidator returns an error object { invalidEmail: true } for an email address with a missing domain part. | Create a mock control with an invalid email address ('invalid@.com'). Call ReferralCustomValidators.emailValidator(control) with the mock control. | The validator should return the error object { invalidEmail: true } when the email address has a missing domain part. |

**Test Purpose:**

This test ensures that the emailValidator function returns an error object when the provided email address has an invalid domain (e.g., @lpl.financial).

**Test Description:**

This test verifies that the emailValidator function correctly identifies an email address with an invalid domain (e.g., invalid@lpl.financial) and returns an error object { invalidEmail: true }.

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 3.13 | emailValidator - Should Return Error Object for an Email Address with an Invalid Domain | Verify that the emailValidator returns an error object { invalidEmail: true } for an email address with an invalid domain (e.g., invalid@lpl.financial). | Create a mock control with an invalid email address ('invalid@lpl.financial'). Call ReferralCustomValidators.emailValidator(control) with the mock control. | The validator should return the error object { invalidEmail: true } for email addresses with invalid domain parts. |

**Test Purpose:**

This test ensures that the emailValidator function returns an error object when the provided email address has extra characters (e.g., multiple @ symbols).

**Test Description:**

This test verifies that the emailValidator function correctly identifies an email address with extra characters (e.g., test@@lplfinancial.com) and returns an error object { invalidEmail: true }.

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 3.14 | emailValidator - Should Return Error Object for an Email Address with Extra Characters | Verify that the emailValidator returns an error object { invalidEmail: true } for an email address with extra characters (e.g., test@@lplfinancial.com). | Create a mock control with an invalid email address ('test@@lplfinancial.com'). Call ReferralCustomValidators.emailValidator(control) with the mock control. | The validator should return the error object { invalidEmail: true } for email addresses with extra characters (like multiple @ symbols). |

**Test Purpose:**

This test ensures that the otherDetailsValidator function returns null for valid input (a string containing alphanumeric characters and not too long).

**Test Description:**

This test verifies that the otherDetailsValidator function correctly validates a valid input string, such as 'Valid details 123', and returns null, indicating no errors.

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 3.15 | otherDetailsValidator - Should Return null for Valid Other Details | Verify that the otherDetailsValidator returns null for valid input, such as a string containing alphanumeric characters. | Create a mock control with valid other details: 'Valid details 123'. Call ReferralCustomValidators.otherDetailsValidator(control) with the mock control. | The validator should return null, indicating that the input is valid and does not violate any validation rules. |

**Test Purpose:**

This test ensures that the otherDetailsValidator function returns an error object when the input string exceeds 50 characters.

**Test Description:**

This test checks that the otherDetailsValidator function correctly identifies invalid input when the provided string is too long (more than 50 characters) and returns an appropriate error object.

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 3.16 | otherDetailsValidator - Should Return Error Object for Invalid Other Details (Too Long) | Verify that the otherDetailsValidator returns an error object when the input exceeds 50 characters. | Create a mock control with an overly long string: 'This is a very long string that should be invalid because it exceeds fifty characters for the test.'. Call ReferralCustomValidators.otherDetailsValidator(control) with the mock control. | The validator should return the error object { invalidOtherDetails: true } for input longer than 50 characters. |

**Test Purpose:**

This test ensures that the otherDetailsValidator function returns an error object when the input contains non-alphanumeric characters (e.g., punctuation marks).

**Test Description:**

This test checks that the otherDetailsValidator function correctly identifies invalid input when the string contains special characters and returns the appropriate error object.

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 3.17 | otherDetailsValidator - Should Return Error Object for Invalid Other Details (Non-Alphanumeric Characters) | Verify that the otherDetailsValidator returns an error object when the input contains non-alphanumeric characters. | Create a mock control with input containing special characters: 'Invalid!@#details'. Call ReferralCustomValidators.otherDetailsValidator(control) with the mock control. | The validator should return the error object { invalidOtherDetails: true } for input containing non-alphanumeric characters. |

**Test Purpose:**

This test ensures that the otherDetailsValidator function returns null for an empty value, indicating that empty input is considered valid.

**Test Description:**

This test checks that the otherDetailsValidator function correctly handles empty input, returning null to signify that empty values are valid.

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 3.18 | otherDetailsValidator - Should Return null for an Empty Value | Verify that the otherDetailsValidator returns null for an empty value, indicating empty input is valid. | Create a mock control with an empty string: ''. Call ReferralCustomValidators.otherDetailsValidator(control) with the mock control. | The validator should return null for the empty input value. |

**Test Purpose:**

This test ensures that the additionalDetailsValidator function returns null for valid input (up to 2000 characters and alphanumeric).

**Test Description:**

This test checks that the additionalDetailsValidator correctly handles valid input, returning null when the input meets the criteria of being alphanumeric and within the character limit.

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 3.19 | additionalDetailsValidator - Should Return null for Valid Additional Details | Verify that the additionalDetailsValidator returns null for valid input (alphanumeric, up to 2000 characters). | Create a mock control with valid additional details: 'Valid additional details up to 2000 characters.'. Call ReferralCustomValidators.additionalDetailsValidator(control) with the mock control. | The validator should return null for valid additional details. |

**Test Purpose:**

This test ensures that the additionalDetailsValidator function returns an error object when the input exceeds the character limit of 2000.

**Test Description:**

This test checks that the additionalDetailsValidator correctly handles inputs that exceed the allowed character limit of 2000, returning an error object.

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 3.20 | additionalDetailsValidator - Should Return Error Object for Invalid Additional Details (Too Long) | Verify that the additionalDetailsValidator returns an error object when the input exceeds 2000 characters. | Create a mock control with a string longer than 2000 characters (e.g., 'A'.repeat(2001)). Call ReferralCustomValidators.additionalDetailsValidator(control) with the mock control. | The validator should return { invalidAdditionalDetails: true } for input that exceeds 2000 characters. |

**Test Purpose:**

This test ensures that the additionalDetailsValidator function returns an error object when the input contains non-alphanumeric characters such as symbols or punctuation marks.

**Test Description:**

This test checks that the additionalDetailsValidator correctly identifies and returns an error object for inputs containing non-alphanumeric characters.

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 3.21 | additionalDetailsValidator - Should Return Error Object for Invalid Additional Details (Non-Alphanumeric Characters) | Verify that the additionalDetailsValidator returns an error object when the input contains special characters. | Create a mock control with input containing special characters like 'Invalid additional details @!#'. Call ReferralCustomValidators.additionalDetailsValidator(control) with the mock control. | The validator should return { invalidAdditionalDetails: true } for input containing non-alphanumeric characters. |

**Test Purpose:**

This test ensures that the additionalDetailsValidator function returns null when the input is empty, indicating that empty input is considered valid.

**Test Description:**

This test checks that the additionalDetailsValidator correctly returns null for an empty string, which is considered valid input.

**Test Cases:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 3.22 | additionalDetailsValidator - Should Return null for an Empty Value | Verify that the additionalDetailsValidator returns null when the input is an empty string. | Create a mock control with an empty string (''). Call ReferralCustomValidators.additionalDetailsValidator(control) with the mock control. | The validator should return null, indicating that empty input is valid. |

**App Component Unit Test**

**Test Purpose:**

This test ensures that the component is created successfully when initialized.

**Test Description:**

This test checks that the component is instantiated without any issues, confirming its successful creation.

**Test Case:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 4.1 | Should Create the App | Verify that the component is created successfully. | 1. Retrieve the component instance using fixture.componentInstance. | The component should be successfully created and the test should pass. Use expect to verify that the component instance is truthy (i.e., the component is created). |

**Test Purpose:**

This test ensures that the title property of the component is correctly set to 'report-unusual-activity'.

**Test Description:**

This test checks that the title property of the component has the correct value.

**Test Case:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 4.2 | Should Have Correct Title | Verify that the title property is set to 'report-unusual-activity' as expected. | 1. Retrieve the component instance using fixture.componentInstance. | The title property should be 'report-unusual-activity', and the test should pass. Use expect to check if component.title equals 'report-unusual-activity'. |

**Test Purpose:**

This test ensures that the methods rest.config and LwkThemes.setTheme are called correctly during the ngOnInit lifecycle hook of the component.

**Test Description:**

The test verifies that the necessary configuration methods (rest.config and LwkThemes.setTheme) are called during the ngOnInit lifecycle of the component.

**Test Case:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 4.3 | Should Call rest.config and LwkThemes.setTheme on ngOnInit | Verify that rest.config and LwkThemes.setTheme methods are called correctly when ngOnInit is triggered. | 1. Create spies for rest.config and LwkThemes.setTheme using jest.spyOn.  2. Call component.ngOnInit(). | Ensure that rest.config is called with the expected configuration and LwkThemes.setTheme is called with the correct theme. Use expect to check that both methods are called. |

**Test Purpose:**

This test ensures that the title property is correctly initialized in the component.

**Test Description:**

The test checks if the title property of the component is set to the expected value during initialization.

**Test Case:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 4.4 | Should Set Title Property Correctly | Verify that the title property is initialized correctly in the component. | After the component is initialized, directly check the title property. | Ensure that the title property is set to 'report-unusual-activity' using expect. |

**App Module Unit Test**

**Test Purpose:**

This test ensures that the AppModule is correctly instantiated and can be created without issues.

**Test Description:**

The test checks if the AppModule is set up properly by attempting to instantiate it and confirming it is truthy.

**Test Case:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 5.1 | Should Create the AppModule | Verify that the AppModule can be instantiated correctly. | Use TestBed.inject(AppModule) to get the instance of the AppModule. | Ensure that the AppModule is truthy using expect. |

**Test Purpose:**

This test ensures that both AppComponent and ReferralComponent can be created successfully and are instantiated correctly.

**Test Description:**

The test verifies if AppComponent and ReferralComponent can be created using Angular's TestBed.createComponent() method and checks if both components are instantiated without errors.

**Test Case:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 5.2 | Should Declare AppComponent and ReferralComponent | Ensure that both AppComponent and ReferralComponent are created correctly. | 1. Create both components using TestBed.createComponent(AppComponent) and TestBed.createComponent(ReferralComponent).  2. Use expect to verify that both components are truthy. | Both components should be instantiated and pass the expect check (i.e., should be truthy). |

**Test Purpose:**

This test ensures that the necessary Angular modules (such as BrowserModule, RouterModule, HttpClientModule, etc.) are imported and available in the testing environment.

**Test Description:**

The test verifies if the essential modules for the application are properly imported and available using Angular's TestBed.inject() method. The test will check that each of these modules is available (truthy), confirming that the setup is correct for the application to run.

**Test Case:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 5.3 | Should Import Necessary Modules | Ensure that the necessary Angular modules are correctly imported. | 1. Use TestBed.inject() to inject required modules like BrowserModule, RouterModule, HttpClientModule.  2. Use expect to check that each module is available (truthy). | The test should pass if all necessary modules are injected and available. |

**Test Purpose:**

This test ensures that the ReferralApiService is correctly provided and available in the testing environment. The goal is to verify that the service can be injected into the components or other services that require it.

**Test Description:**

The test verifies that ReferralApiService is available for injection by retrieving an instance using TestBed.inject() and confirming that the service is correctly provided in the testing environment.

**Test Case:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 5.4 | Should Provide ReferralApiService | Ensure that the ReferralApiService is correctly provided and available. | 1. Use TestBed.inject(ReferralApiService) to retrieve an instance of the service.  2. Use expect to verify that the service instance is truthy. | The test should pass if the ReferralApiService is provided and available for injection. |

**App Route Module**

**Test Purpose:**

This test ensures that the AppRoutingModule is correctly instantiated in the testing environment. It checks whether the routing module is successfully created and available for use in the app.

**Test Description:**

The test verifies that the AppRoutingModule can be injected into the testing environment using TestBed.inject(), confirming that the routing module is properly set up and available.

**Test Case:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 6.1 | Should Create the AppRoutingModule | Ensure that the AppRoutingModule is correctly created. | 1. Use TestBed.inject(AppRoutingModule) to retrieve an instance of the module.  2. Use expect() to assert that the module is truthy. | The test should pass, confirming that the AppRoutingModule is correctly instantiated. |

**Test Purpose:**

This test ensures that the routes in the app's routing configuration are correctly defined and match the expected paths and components.

**Test Description:**

The test verifies that the application's routes are correctly set up by checking that the path and the associated component for each route are as expected.

**Test Case:**

| **S.No.** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 6.2 | Should Have Correct Routes Configured | Ensure that the routes in the appRoutes array are correctly configured. | 1. Define the expected routes: Routes = appRoutes.  2. Use expect() to verify the routes. | The routes should match the following configuration:   **Route 1**: /report-suspicious-activity/:account\_no should route to ReferralComponent.   **Route 2**: /referral/:account\_no should route to ReferralComponent.   **Route 3**: /referral should route to ReferralComponent. |

**Test Purpose:**  
This test ensures that the RouterModule is correctly imported and exported within the AppRoutingModule.

**Test Description:**  
This test checks if the RouterModule is correctly injected and available in the testing environment, confirming that the routing module is set up correctly in AppRoutingModule.

| **S.No** | **Test Case For** | **Scenario** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| 6.3 | Import and Export RouterModule | Check if AppRoutingModule imports RouterModule | Use TestBed.inject(AppRoutingModule) | Verify if AppRoutingModule is correctly instantiated. |
|  | Inject RouterModule | Ensure RouterModule is correctly injected | Use TestBed.inject(RouterModule) | Verify that RouterModule is correctly injected into the testing environment. |
|  | Check Injection Availability | Ensure that both modules are truthy | N/A | Both AppRoutingModule and RouterModule should be truthy or defined. |