### **Test Plan**

#### **1. Introduction**

This test plan outlines the testing strategy for the commercial registration verification process via third-party service [X]. The purpose is to ensure the system accurately verifies the commercial registration number, validates the data, and handles various scenarios effectively.

#### **2. Scope**

The scope includes:

* Integration with third-party service [X].
* Verification of valid and invalid commercial registration numbers.
* Validation of company start dates for eligibility.
* Appropriate notifications and error handling for different scenarios.

#### **3. Objectives**

* Verify the integration with [X] to validate commercial registration numbers.
* Ensure valid commercial registration numbers allow the user to proceed.
* Confirm that invalid numbers prompt users to re-enter and restart the verification process.
* Validate start date logic to enforce the two-year eligibility rule.

#### **4. Approach**

* **Functional Testing**: Validate the integration with [X] and the system's response to valid and invalid data.
* **Boundary Testing**: Test edge cases for start date validation.
* **Negative Testing**: Check system behavior with invalid or incomplete data.
* **Integration Testing**: Ensure smooth communication between the system and [X].
* **User Experience Testing**: Validate notifications and user flow for clarity and usability.

#### 

#### 

#### 

#### **5. Test Items**

The following features will be tested:

1. Sending the commercial registration number to [X].
2. Handling valid and invalid responses from [X].
3. Validating company start dates.
4. Restarting the verification process for invalid numbers.
5. Notifying users of eligibility based on start date validation.

#### **6. Test Environment**

* **Operating Systems**: Windows, macOS, Linux.
* **Browsers**: Chrome, Firefox, Safari, Edge (latest versions).
* **Backend Integration**: Test server connected to [X].
* **Database**: Test database with mock data for commercial registrations.
* **Tools**: Postman (API testing), JIRA (bug tracking), TestRail (test management).

#### **7. Roles and Responsibilities**

* **QA Engineers**: Design and execute test cases, report defects, and validate fixes.
* **Developers**: Address defects and assist in integration testing.
* **Test Manager**: Oversee test execution and report progress.
* **Stakeholders**: Review test results and approve deployment.

#### **8. Test Deliverables**

* Test Cases
* Test Execution Results
* Defect Reports
* Test Summary Report

#### **9. Entry Criteria**

* Integration with [X] is complete and functional.
* Test data (valid and invalid commercial registration numbers) is prepared.
* Test environment is set up.

#### **10. Exit Criteria**

* All critical test cases executed.
* No unresolved high-priority defects.
* Stakeholders approve the test summary report.

#### **11. Risks**

| **Risk** | **Mitigation Strategy** |
| --- | --- |
| Integration with [X] fails during testing. | Use mock APIs for testing until [X] is available. |
| Incorrect validation of start date data. | Perform detailed validation and cross-check outputs. |
| Delay in response from third-party [X]. | Implement retries and timeout handling. |

#### **12. Test Schedule**

| **Phase** | **Duration** | **Activities** |
| --- | --- | --- |
| Test Planning | 3 days | Prepare test cases and data. |
| Test Execution | 5 days | Execute tests and log defects. |
| Bug Fixing and Retesting | 2 days | Retest fixed defects. |
| Test Reporting | 2 days | Prepare and share test summary report. |

#### **13. Test Cases**

##### **Scenario 1: Valid Data**

* **TC01**: Verify the system sends the commercial registration number to [X].
* **TC02**: Verify the system receives a valid response from [X].
* **TC03**: Verify the user can proceed to the next step after a valid response.

##### **Scenario 2: Invalid Data**

* **TC04**: Verify the system handles an invalid registration number response.
* **TC05**: Verify the system prompts the user to re-enter the registration number.
* **TC06**: Verify the system restarts the verification process with the new number.

##### **Scenario 3: Validation for Start Date**

* **TC07**: Verify the system validates the start date received from [X].
* **TC08**: Verify the system notifies the user if the start date is less than two years.
* **TC09**: Verify the system allows the user to proceed if the start date is more than two years.

##### **Edge Cases**

* **TC10**: Verify system behavior if [X] service is unavailable.
* **TC11**: Verify handling of extremely large or small registration numbers.
* **TC12**: Verify system behavior with unexpected responses from [X].

#### **14. Test Runs**

1. **Test Run 1: Happy Path  
   Objective**: Verify successful verification of valid commercial registration numbers.  
   **Test Cases**: TC01, TC02, TC03.
2. **Test Run 2: Negative Scenarios  
   Objective**: Test system behavior with invalid registration numbers.  
   **Test Cases**: TC04, TC05, TC06.
3. **Test Run 3: Validation Testing  
   Objective**: Verify start date validation logic.  
   **Test Cases**: TC07, TC08, TC09.
4. **Test Run 4: Edge Cases  
   Objective**: Test uncommon or failure scenarios, such as service unavailability.  
   **Test Cases**: TC10, TC11, TC12.

#### **15. Conclusion**

This test plan ensures thorough testing of the corporate investor commercial registration verification process via [X]. Successful execution will confirm the system's reliability and usability for corporate investors.

### **Risk-Based Testing (RBT) Plan**

#### **Risk Analysis**

| **Risk ID** | **Risk** | **Impact** | **Likelihood** | **Priority** | **Mitigation Strategy** |
| --- | --- | --- | --- | --- | --- |
| R1 | Integration with third-party service [X] fails during verification. | High | Medium | Critical | Use mock APIs to simulate [X] responses until live integration is verified. |
| R2 | Validation logic for start date is incorrectly implemented. | High | High | Critical | Add detailed test cases for boundary conditions and cross-check against expected results. |
| R3 | User-facing error messages are unclear or misleading. | Medium | High | High | Validate all user notifications and ensure clarity through User Acceptance Testing (UAT). |
| R4 | System fails to handle invalid commercial registration numbers gracefully. | Medium | Medium | High | Add test cases to validate all possible invalid data inputs and system responses. |
| R5 | Unexpected downtime or unavailability of third-party service [X]. | High | Medium | High | Implement retry logic and fallback mechanisms for third-party API calls. |
| R6 | Notification or alert delivery (e.g., SMS) fails. | Medium | Medium | Medium | Test edge cases for notification delivery and ensure error handling for failed SMS gateway scenarios. |

### 

### **Test Cases for Each User Story**

#### **User Story: Validating Commercial Registration via [X]**

### **Scenario 1: Valid Data**

| **Test Case ID** | **Test Scenario** | **Test Steps** | **Expected Result** |
| --- | --- | --- | --- |
| **TC-01** | Verify the system sends the commercial registration number to [X]. | 1. Submit a valid commercial registration number.  2. Check if the number is sent to [X]. | The system sends the registration number to [X] successfully. |
| **TC-02** | Verify the system receives a valid response from [X]. | 1. Submit a valid registration number.  2. Check the response from [X]. | The system receives a valid response from [X]. |
| **TC-03** | Verify the system allows the user to proceed after a valid response. | 1. Submit a valid registration number.  2. Receive valid response.  3. Proceed to validation. | The user is allowed to proceed to the validation process. |

### 

### 

### **Scenario 2: Invalid Data**

| **Test Case ID** | **Test Scenario** | **Test Steps** | **Expected Result** |
| --- | --- | --- | --- |
| **TC-04** | Verify the system handles an invalid commercial registration number response from [X]. | 1. Submit an invalid registration number.  2. Check the response from [X]. | The system identifies the invalid registration number. |
| **TC-05** | Verify the user is prompted to re-enter the registration number upon receiving an invalid response. | 1. Submit an invalid registration number.  2. Check the system prompt for re-entry. | The system prompts the user to re-enter the registration number. |
| **TC-06** | Verify the system restarts the verification process with a new registration number. | 1. Submit an invalid registration number.  2. Re-enter a valid registration number.  3. Restart process. | The system restarts the verification process with the newly entered registration number. |

### 

### 

### **Scenario 3: Validation for Start Date**

| **Test Case ID** | **Test Scenario** | **Test Steps** | **Expected Result** |
| --- | --- | --- | --- |
| **TC-07** | Verify the system validates the company start date provided by [X]. | 1. Submit a registration number.  2. Receive valid response from [X].  3. Check the start date. | The system validates the start date provided by [X]. |
| **TC-08** | Verify the system notifies the user if the start date is less than two years. | 1. Submit a registration number.  2. Receive response with start date < 2 years.  3. Check the notification. | The system notifies the user they cannot proceed due to a start date of less than two years. |
| **TC-09** | Verify the system allows the user to proceed if the start date is more than two years. | 1. Submit a registration number.  2. Receive response with start date > 2 years.  3. Proceed to next step. | The system allows the user to proceed to the next step after validating the start date. |

### **Edge Cases**

| **Test Case ID** | **Test Scenario** | **Test Steps** | **Expected Result** |
| --- | --- | --- | --- |
| **TC-10** | Verify the system handles timeouts or unavailability of [X]. | 1. Submit a registration number.  2. Simulate [X] being unavailable.  3. Check system behavior. | The system provides an appropriate error message for unavailability of [X]. |
| **TC-11** | Verify the system processes unexpected responses or errors from [X]. | 1. Submit a registration number.  2. Simulate an unexpected response from [X].  3. Check system response. | The system handles unexpected responses gracefully and logs the error. |
| **TC-12** | Verify the system processes extremely large or invalid registration numbers gracefully. | 1. Submit a very large or invalid registration number.  2. Check system behavior. | The system handles the input gracefully, providing error messages or restrictions as needed. |

### 

### 

### 

### 

### 

### **Test Runs**

#### **Test Run 1: Happy Path**

**Objective**: Ensure successful verification of valid commercial registration numbers and start dates.  
**Test Cases**: TC01, TC02, TC03, TC09.

#### **Test Run 2: Negative Scenarios**

**Objective**: Test the system's behavior when encountering invalid or unexpected data.  
**Test Cases**: TC04, TC05, TC06, TC08.

#### **Test Run 3: Edge Case Testing**

**Objective**: Test uncommon scenarios, including timeouts and unexpected responses from [X].  
**Test Cases**: TC10, TC11, TC12.

#### **Test Run 4: Boundary Testing**

**Objective**: Validate start date boundary conditions for less than and more than two years.  
**Test Cases**: TC07, TC08, TC09.

#### **Test Run 5: Integration and Notification Testing**

**Objective**: Ensure smooth integration with [X] and verify the accuracy of notifications.  
**Test Cases**: TC01, TC02, TC05, TC06.