Selenium QA Assignment: Automated Web Testing Challenge

As a QA engineer, your task is to automate the following user journey on a website using **Selenium WebDriver** and **Python**.

Scenario:

You are given a task to verify the login functionality and form interaction on a website. The goal is to:

1. Navigate to a specified website.
2. Verify that certain elements are present and interactable.
3. Perform login using provided credentials.
4. Verify successful login.
5. Perform a search using a keyword.
6. Verify that search results are displayed.
7. Capture a screenshot for both success and failure cases.

Detailed Task Breakdown:

1. Navigate to the Website:

- Open the browser (Firefox) and navigate to `https://inc42.com/`.
- Maximize the browser window for better visibility.

2. Verify the Presence of Key Elements:

- Verify that a **"Datalabs"** link is present on the homepage and that it is clickable. This link is usually in the navigation menu at the top.
- Once found, click the **"Datalabs"** link to proceed.

3. Login to the Website:

- On the Datalabs page, find the **"Login"** button.
- Verify that the login button is visible and clickable.
- Once clicked, enter the following login credentials in the form:
    - **Email**: `inc@yopmail.com`
    - **Password**: `Vishal@inc42`
- Click the submit button to attempt login.

4. Verify Successful Login:

- After submitting the login form, verify that the login is successful by checking for a **dashboard element** or any user-specific content that is only visible after login.
- If the login is successful, capture a screenshot as **"login_successful.png"**.

5. Perform a Search:

- After logging in, locate the **search bar** on the dashboard or main page.
- Input the keyword **"Fintech"** into the search bar and submit the search.
- Verify that the search results are displayed.

6. Verification of Search Results:

- Ensure that relevant search results are displayed for the keyword **"Fintech"**.
- Capture a screenshot of the results as **"search_results.png"**.

7. Error Handling & Screenshots:

- Implement proper error handling. If any step fails (e.g., login failure, missing elements), capture a screenshot with the failure reason, such as **"login_failed.png"** or **"search_failed.png"**.

Requirements:

- **Use Selenium WebDriver**: You must use Selenium WebDriver for browser automation.
- **Implement Waits**: Use **explicit waits** (`WebDriverWait`) for interacting with dynamic elements to avoid timing issues.
- **Screenshots**: Ensure you capture screenshots for both success and failure scenarios.
- **Error Handling**: Implement basic error handling (try-except) to ensure that errors are logged properly and the browser closes gracefully, even when there's an issue.

Deliverables:

1. **Python script** that automates the above task.
2. **Screenshots** showing successful login and search results.
3. **Logs** or error messages in case of failure (e.g., elements not found).

Use Selenium WebDriver: You must use Selenium WebDriver for browser automation.
Implement Waits: Use explicit waits (WebDriverWait) for interacting with dynamic elements to avoid timing issues.

Screenshots: Ensure you capture screenshots for both success and failure scenarios.

Error Handling: Implement basic error handling (try-except) to ensure that errors are logged properly and the browser closes gracefully, even when there's an issue.

Deliverables:

- Python script that automates the above task.
- Screenshots showing successful login and search results.
- Logs or error messages in case of failure (e.g., elements not found).