# Contact List Application - QA Assignment

## 1. Test Plan

This test plan outlines the strategy for testing the Contact List application. The application allows users to register, log in, and manage a contact list through UI and API. The testing objectives are to ensure the critical functionalities work as expected and identify any defects.

### Scope

The scope of testing includes:  
1. User registration and login functionalities.  
2. CRUD operations (Create, Read, Update, Delete) on the contact list.  
3. API testing for endpoints related to user and contact management.  
4. UI testing for user flows and validation.

### Testing Approach

1. Functional testing to verify features against requirements.  
2. Smoke testing to validate critical workflows.  
3. Positive and negative testing to check for valid and invalid scenarios.  
4. Automation of smoke tests using Python.

## 2. UI and API Test Cases

Below are the detailed test cases for the application, focusing on real user flows and combining multiple API calls where applicable.

### UI Test Cases

| Test Case ID | Description | Steps & Result |
| --- | --- | --- |
| UI-01 | User Login | Steps: 1. Navigate to Register page. 2. Enter invalid details. 3. Click 'Register'.  Result: User is redirected to Login page with a “Incorrect username or password” message. |
| UI-02 | User Login | Steps: 1. Navigate to Login page. 2. Empty credentials. 3. Click 'Login'.  Result: User is redirected to Login page with a “Incorrect username or password” message. User is redirected to the Contact List page. |
| UI-03 | User Registration | Steps: 1. Navigate to Register page. 2. Click 'Sign Up'. 3. Fill User details that does’t exhist. 4. Click ‘Submit’.  Result: Redirect to “Contact List” and being Logged in. |
| UI-04 | Contact Creation | Steps: 1. Navigate to Contact List page. 2. Click 'Add a New Contact'. 3. Fill Contact details.. 4. Click ‘Submit’.  Result: Redirect to “Contact List” and where you can find new added contact. |
| UI-05 | Edit Contact | Steps: 1. Navigate to Contact List page. 2. Click on Contact row. 3. Click on Edit Contact. 4. Make changes in contact details.  5. Click Submit  Result: Redirect to “Contact List” after saving Contact data. |
| UI-06 | Delete Contact | Steps: 1. Navigate to Contact List page. 2. Click on Contact row. 3. Click on Delete Contact. 4. Confirm by clicking “OK” when alert appears.  Result: Redirect to “Contact List” after deleting Contact data. |

### API Test Cases

| Test Case ID | Endpoint | Description | Request & Expected Response |
| --- | --- | --- | --- |
| API-1 | /users | Register a user with valid data | Request: POST { 'email': 'test@example.com', 'password': 'password123' }  Expected Response: Status 201, message 'User registered successfully'. |
| API-2 | /users/login | Login with valid credentials | Request: POST { 'email': 'test@example.com', 'password': 'password123' }  Expected Response: Status 200, token returned. |
| API-3 | /contacts | Fetch contact list after login | Request: GET with Authorization Header.  Expected Response: Status 200, list of contacts. |
| API-4 | /contacts | Add a new contact | Request: POST {'firstName': 'Sinan','lastName': 'Sakic', 'email': 'sinansakic@example.com', 'phone': '123456789' } with Auth Header.  Expected Response: Status 201, contact added. |
| API-5 | /contacts/[contact\_id] | Edit Contact | Request: PUT{ "\_id": "608b2db1add2691791c04c89",'firstName': 'Sinan','lastName': 'Sakic', 'email': 'sinansakic@example.com', 'phone': '123456789' } with Auth Header.  Expected Response: Status 200, contact added. |
| API-6 | /contacts/[contact\_id] | Delete Contact | Request: DELETE with Authorization Header.  Expected Response: Status 200, list of contacts. |

## 3. Smoke Tests

The following smoke tests were identified:  
1. User login (UI and API).  
2. Fetch contact list (API).  
3. Add a new contact (UI and API).  
4. Edit and delete (UI and API).

## 4. Automated Smoke Tests

Automated smoke tests were implemented in Python. Below are the instructions to run the tests:  
1. Install Python dependencies using `pip install selenium requests`.  
2. Run `smoke\_ui\_test.py` for UI tests.  
3. Run `smoke\_api\_test.py` for API tests.

## 5. Bug Reporting

**1. Issue: Name and Surname Fields During User Creation**

**Description:**

The name and surname fields during the user creation process allow any text input, with no validation on the format. This could lead to inconsistencies in how names are displayed on the platform.

**Steps to Reproduce:**

1. Navigate to the user creation form.
2. Input any random text in the name and surname fields(e-mail is required).
3. Proceed with the account creation.

**Expected Result:**

Name and surname fields should follow a specific format or validation rules (e.g., only alphabetic characters, proper casing).

**Actual Result:**

The fields allow any kind of text, and there is no restriction on input format.  
  
**2. Issue: No Date Picker for Birthdate Field**

**Description:**The birthdate field does not have a date picker, requiring the user to manually type the date, which can lead to formatting errors or inconsistencies.

**Steps to Reproduce:**

1. Navigate to the user creation form.
2. Attempt to input a birthdate manually without a calendar or date picker.

**Expected Result:**A date picker should be available to simplify the process of selecting a birthdate.

**Actual Result:**The birthdate field requires manual entry, and there is no date picker functionality available.

### 3. Issue: Address, City, and Country Fields Not Displaying Commas

**Description:**When entering an address, city, and country, the form does not automatically insert commas between the values, even though it seems to be designed for such formatting.

**Steps to Reproduce:**

1. Navigate to the user creation form.
2. Enter an address, city, and country without commas.
3. Submit the form.

**Expected Result:**The address, city, and country fields should display commas between the values automatically or allow users to manually input them.

**Actual Result:**The address, city, and country fields do not display commas automatically between them.

### 4. Issue: Website Pages Accessible Without Login

**Description:**The website allows users to access pages without logging in, which can lead to unauthorized access to content and other potential security risks.

**Steps to Reproduce:**

1. Open the website.
2. Navigate to a page that should require login.
3. Observe that the page is accessible without authentication.

**Expected Result:**Users should be redirected to a login page when attempting to access restricted pages without being logged in.

**Actual Result:**Pages can be accessed without logging in.

**5. Issue: No Session Management for Logged-In Users**

**Description:**The website does not maintain session management for logged-in users, meaning that when a user is logged in and opens the website in a new tab, they are redirected to the login page instead of being automatically logged in.

**Steps to Reproduce:**

1. [Log in](https://thinking-tester-contact-list.herokuapp.com/) to the website.
2. Open a new tab and navigate to the website.
3. Observe that the user is presented with the login page instead of being automatically logged in.

**Expected Result:**The website should retain the login state across tabs and should automatically log the user in when they open a new tab.

**Actual Result:**There is no session management; the user is directed to the login page when opening the site in a new tab.

### 6. Issue: Multiple Logins and Data Syncing

**Description:**The website allows users to log in with two accounts simultaneously, but it only displays data from the last logged-in account. Upon refreshing the tab, the data reflects the correct account information.

**Steps to Reproduce:**

1. [Log in](https://thinking-tester-contact-list.herokuapp.com/) with two different accounts in separate tabs.
2. Switch between the tabs and observe the displayed data.
3. Refresh one of the tabs.

**Expected Result:**Each tab should reflect the correct user data for the account logged into that tab, without interference from other sessions.

**Actual Result:**Only the data from the last logged-in account is displayed across tabs.  
  
**7. Issue: Email Address Already in Use Throws 400 Bad Request**

**Description:**When attempting to sign up with an email that already has an account, the website throws a 400 Bad Request error in console.

**Steps to Reproduce:**

1. Attempt to [sign up](https://thinking-tester-contact-list.herokuapp.com/addUser) with an email that is already registered.
2. Observe the error message.

**Expected Result:**The website shouldn’t display any error in console indicating that the email is already in use.

**Actual Result:**A 400 Bad Request error is returned in console when attempting to sign up with an already registered email.

### 8. Issue: Option to Log Out When Not Logged In

**Description:**When trying to enter a random page without being logged in, the website displays an option to log out, even though the user is not logged in.

**Steps to Reproduce:**

1. Navigate to a [random page](https://thinking-tester-contact-list.herokuapp.com/atlant) without logging in.
2. Observe the presence of the log-out option.

**Expected Result:**The log-out option should not be displayed if the user is not logged in.

**Actual Result:**The log-out option is visible even when the user is not logged in.

**9. Issue: No Validation on First Name and Last Name During Contact Creation  
Description:**During contact creation, the first name and last name fields allow any input without validation, meaning users can enter any value regardless of format or content.

**Steps to Reproduce:**

1. Navigate to the [contact creation page](https://thinking-tester-contact-list.herokuapp.com/addContact).
2. Enter random or invalid values in the first name and last name fields.
3. Submit the form.

**Expected Result:**The first name and last name fields should have proper validation to ensure correct format (e.g., only alphabetic characters, proper casing).

**Actual Result:**The first name and last name fields allow any input without restrictions or validation.