Odine Freelance Marketplace: QA Test Approach

1. Overview

This document outlines the QA test approach for the Odine Freelance Marketplace application, based on its functional and non-functional requirements. The focus is on leveraging the available mock data from JSONPlaceholder to ensure comprehensive testing while addressing potential limitations.

---

2. Test Scenarios

2.1 Dashboard Test Scenarios

1. **Freelancer List Loading**
   - Ensure the dashboard displays the freelancer list with the following attributes:
     - Name
     - Email
     - Phone
     - City
     - Finished Job Count
   - Verify the mock photo placeholders load correctly.
2. **Search by Name**
   - Validate the search dynamically filters freelancers based on partial or full matches of their name.
3. **Search by Finished Job Count**
   - Test filtering freelancers by specifying a range (e.g., 1-10) for job counts.
4. **Search by City**
   - Confirm the filtering functionality works correctly based on the city selected or typed.
5. **Light/Dark Mode Toggle**
   - Verify that switching between light and dark modes updates the UI instantly and retains user preferences.
6. **Save Freelancer**
   - Test adding freelancers to the saved list and verify the saved list displays correctly when the "Saved Freelancers" filter is applied.

2.2 Portfolio Page Test Scenarios

1. **Portfolio Page Loading**
   - Verify freelancer details are displayed correctly, including:
     - Name
     - Phone
     - Website
     - Address
   - Ensure the job list fetched from the `/posts` endpoint is correctly displayed.

2. **Show Comments Functionality**
    - Confirm the "Show Comments" button fetches and displays comments for a selected job from the `/comments` endpoint.
3. **Hire Freelancer Popup**
    - Validate the popup form contains fields for name, subject, and message, and that submission is simulated successfully.

2.3 Hire Freelancer Popup Test Scenarios

1. **Popup Form Opening**
    - **Purpose:** To ensure that the Hire Freelancer popup opens correctly.
    - **Steps:**
        1. Click the "Hire Freelancer" button on the freelancer's profile page.
        2. Verify that the popup form opens correctly.
    - **Expected Result:**
        - All required fields ("Name", "Message Subject", "Message Body") should be displayed.
        - Ensure the form is ready for user interaction.
2. **Correct Display of Form Fields**
    - **Purpose:** To verify that all the form fields are displayed correctly.
    - **Steps:**
        1. Click the "Hire Freelancer" button.
        2. Ensure that the "Name", "Message Subject", and "Message Body" fields are correctly displayed in the popup.
    - **Expected Result:**
        - All three fields should be visible and user-interactive.
3. **Form Submission with Valid Data**
    - **Purpose:** To verify that the form can be submitted with valid data.
    - **Steps:**
        1. Enter a valid name in the "Name" field.
        2. Enter a valid subject and message in the "Message Subject" and "Message Body" fields.
        3. Click the "Submit" button.
    - **Expected Result:**
        - The form data should be processed correctly, and the popup should close or show a success message (since real data submission is simulated).
4. **Form Submission with Invalid Data**
    - **Purpose:** To ensure that appropriate error messages are shown when invalid or missing data is provided.
    - **Steps:**
        1. Leave the "Name", "Message Subject", and "Message Body" fields empty.
        2. Click the "Submit" button.
    - **Expected Result:**
        - Appropriate error messages should appear for each empty field, and the form submission should be prevented.

5. **Simulated Form Submission**
    - **Purpose:** To simulate the form submission without sending data to the backend.
    - **Steps:**
        1. Fill in the form with valid data.
        2. Click the "Submit" button.
    - **Expected Result:**
        - A simulated confirmation message or a "Form submitted" message should appear, indicating the form submission process.

2.4 Dark/Light Mode Switching Test Scenarios

1. **Ability to Switch Modes**
    - **Objective:** Ensure that the user can switch between dark and light modes.
    - **Steps:**
        1. Open the application, which should be in light mode by default.
        2. Click on the toggle to switch to dark mode.
        3. Verify that the application immediately switches to dark mode.
        4. Switch back to light mode and verify the mode change again.
    - **Expected Result:**
        - The mode should switch instantly between light and dark, and the interface should update accordingly.
2. **Mode Persistence Across Sessions**
    - **Objective:** Ensure that the selected mode (dark or light) persists across user sessions.
    - **Steps:**
        1. Open the app in light mode.
        2. Switch to dark mode.
        3. Close the app and reopen it.
        4. Verify that the mode (dark or light) persists when the app is reopened.
    - **Expected Result:**
        - The selected mode should persist after the app is closed and reopened, retaining the last selected mode.
3. **UI and Performance During Mode Change**
    - **Objective:** Ensure that switching between modes does not affect UI layout or performance.
    - **Steps:**
        1. Open the application in light mode and interact with different UI elements (buttons, menus, etc.).
        2. Switch to dark mode.
        3. Re-interact with the same UI elements in dark mode.
    - **Expected Result:**
        - The application should maintain its functionality and performance in both modes without any issues. UI elements should render correctly in both light and dark modes.

2.5 Save Freelancer Functionality Test Scenarios

1. **Save Freelancer Functionality**
    - **Objective:** Verify that users can save a freelancer to their favorites.
    - **Steps:**
        1. Go to a freelancer's profile page.
        2. Click on the "Save Freelancer" button.
        3. Verify that a confirmation message appears indicating the freelancer has been saved to favorites.
    - **Expected Result:**
        - The freelancer should be saved successfully, and a confirmation message should be displayed.
2. **View Saved Freelancers**
    - **Objective:** Ensure that users can view saved freelancers using the "Saved Freelancers" filter.
    - **Steps:**
        1. Save a freelancer by clicking on the "Save Freelancer" button.
        2. Apply the "Saved Freelancers" filter on the dashboard or relevant page.
        3. Verify that the saved freelancer appears in the filtered list.
    - **Expected Result:**
        - The saved freelancer should be listed under the "Saved Freelancers" filter.
3. **Remove Saved Freelancer**
    - **Objective:** Verify that users can remove a saved freelancer from their favorites.
    - **Steps:**
        1. Go to the "Saved Freelancers" section.
        2. Click on the "Unsave Freelancer" button next to a saved freelancer.
        3. Verify that the freelancer is removed from the list of saved freelancers.
    - **Expected Result:**
        - The "Unsave Freelancer" button should remove the freelancer from the saved list, and they should no longer appear in the "Saved Freelancers" filter.
4. **Freelancer Removal from Saved List and Visibility**
    - **Objective:** Verify that removing a freelancer from the saved list doesn't affect their appearance in the regular list of freelancers.
    - **Steps:**
        1. Save a freelancer to favorites.
        2. Remove the freelancer from the saved list by clicking the "Unsave" button.
        3. Go back to the main freelancer list and verify that the freelancer still appears in the normal list.
    - **Expected Result:**
        - The freelancer should be removed from the "Saved Freelancers" filter but should remain visible in the regular freelancer list.

Ömer Örnek