**Test Scenarios:**

1. **Page Load Verification**

Scenario: Validate successful page loading without errors.

Objective: Confirm basic functionality and ensure a smooth user experience.

Outcome: Pass if the page loads seamlessly; fail otherwise.

**2.** **API Endpoint Compliance**

Scenario: Verify adherence of each API endpoint to specified collections.

Objective: Ensure that each endpoint returns the expected data and follows the defined specifications.

Outcome: Pass if all API endpoints meet specifications; fail otherwise.

**3. Data Consistency Validation**

Scenario: Check if displayed data on each page corresponds to the data retrieved from the respective API endpoint.

Objective: Guarantee data consistency between the frontend and the API.

Outcome: Pass if data consistency is maintained; fail otherwise.

**Recommendations for Improvement:**

1. Effective Error Handling

Recommendation: Implement robust error-handling mechanisms.

Rationale: Enhance code resilience and ease issue identification during development and testing.

Action: Employ structured error-handling using try-catch blocks.

2. Version Control Integration

Recommendation: Integrate version control with a secure Git repository.

Rationale: Enhance collaboration, track changes, and provide a safety net for code management.

Action: Set up a private Git repository and commit changes regularly.

3. Syntax Refinement

Recommendation: Rectify syntax discrepancies in the existing code.

Rationale: Ensure smooth code execution by addressing syntax issues.

Action: Correct the syntax errors present in the provided code.

4. Enhanced Logging Implementation

Recommendation: Improve logging practices for comprehensive debugging and monitoring.

Rationale: Detailed logs facilitate better comprehension of code flow and effective issue identification.

Action: Enhance logging statements for improved clarity in error tracing.