Habib University

Dhanani School of Science and Engineering

CS/CE 353/374-L2 Software Engineering

Testing Strategy

SHOQ Mobiles

Group Members

Hammad Sajid (hs07606)

Sarim Tahir (st07112)

Owais Waheed (ow07611)

Muhammad Khubaib (mk07218)

Contents

1 Introduction

This document outlines the testing approach, strategies, and frameworks used for the mobile data scraping web application. The application aims to scrape mobile data from various e-commerce websites, provide filtering based on prices and models, and redirect users to the main page when a mobile is selected. It requires no login and includes functionalities such as search, view, filter, redirect, and data scraping from different e-commerce online stores.

2 Backend Testing

2.1 API Testing

The backend APIs will be tested using a combination of unit tests and integration tests. Unit tests will cover individual functions and routes, while integration tests will ensure the correct behavior of the API when multiple components interact.

2.1.1 Tools and Libraries

- **Jest**: A JavaScript testing framework for writing and running tests.
- **Supertest**: A library for testing Node.js HTTP servers.

2.1.2 Test Suite Structure

- `api/routes/`:
  - `scraper.test.js`: Tests for the scraping routes.
  - `filter.test.js`: Tests for the filtering routes.
  - `redirect.test.js`: Tests for the redirection routes.
- `utils/`:
  - `scraperHelpers.test.js`: Tests for the scraping utility functions.
  - `filterHelpers.test.js`: Tests for the filtering utility functions.

2.2 Unit Testing

Unit tests will cover individual functions and utility modules used in the backend codebase.

2.2.1 Tools and Libraries

- **Jest**: A JavaScript testing framework for writing and running tests.

2.2.2 Test Suite Structure

- `utils/`:
  - `scraping.test.js`: Tests for scraping functions.
  - `filtering.test.js`: Tests for filtering functions.

2.3 Integration Testing

Integration tests will ensure the correct behavior of the backend components when multiple modules interact.

2.3.1 Tools and Libraries

- **Jest**: A JavaScript testing framework for writing and running tests.
- **Supertest**: A library for testing Node.js HTTP servers.

2.3.2 Test Suite Structure

- `integration/`:
  - `scraping.test.js`: End-to-end tests for the scraping flow.
  - `filtering.test.js`: End-to-end tests for the filtering flow.
  - `redirect.test.js`: End-to-end tests for the redirection flow.

2.4 Example

- `Scraping API`

```
    // scraper.test.js
const request = require('supertest');
const app = require('../server'); // Adjust path as needed

describe('Scraping API - /scrape', () => {
    it('should scrape mobile data from e-commerce websites', async () => {
        // Test implementation here
    });
});
```

3 Frontend Testing

3.1 Unit Testing

Unit tests will cover individual components, hooks, and utility functions used in the Flutter frontend codebase.

3.1.1 Tools and Libraries

- **Flutter test package**: Testing package provided by Flutter for writing and running tests.

3.1.2 Test Suite Structure

- `test/`:
  - `mobile_list_test.dart`: Tests for the mobile list widget.
  - `filter_test.dart`: Tests for the filtering functionality.

3.2 Integration Testing

Integration tests will ensure the correct behavior of the frontend components when multiple components interact. These tests will simulate real-world scenarios and cover end-to-end flows.

3.2.1 Tools and Libraries

- **Flutter test package**: Testing package provided by Flutter for writing and running tests.

3.2.2 Test Suite Structure

- `test/`:
  - `end_to_end_test.dart`: End-to-end tests for the entire application flow.

4 Test Plan Execution

4.1 Test Preparation

- Set up test environments and tools.
- Prepare test data and scenarios.

4.2 Test Execution

- Execute test cases according to the test plan.
- Record test results and any issues encountered.

4.3 Review and Analysis

- Review test results and identify areas for improvement.
- Analyze defects and prioritize them for resolution.

4.4 Test Deliverables

- Test Execution Reports
- Defect Reports

4.5 Test Environment

- Software Environment: Web browsers (Chrome)
- Devices: Desktop, laptop, and mobile devices
- Testing Tools: Test management tool, Bug tracking tool

5 Test Cases

5.1 Searching for Products

TC-1: Valid Search Keyword Entry:

- **Input**: User enters a valid keyword in the search bar.
- **Expected Output**: Relevant product listings related to the keyword are displayed.

TC-2: Valid Filter Application:

- **Input**: User applies filters such as category, price range, and brand.
- **Expected Output**: Product listings are filtered based on the applied filters, and only relevant products are displayed.

TC-3: Empty Search Keyword Entry:

- **Input**: User submits the search form without entering any keyword.
- **Expected Output**: User is prompted to enter a keyword, and no products are displayed until a valid keyword is entered.

5.2 Viewing Product Listings

TC-4: Detailed Product Listing Display:

- **Input**: User clicks on a product to view its details.
- **Expected Output**: Detailed information about the product including images, descriptions, prices, and seller information are displayed.

TC-5: Clear and Organized Presentation:

- **Input**: User navigates through multiple product listings.
- **Expected Output**: Product listings are presented in a clear and organized manner, facilitating easy comparison between products.

5.3 Comparing Prices and Deals

TC-6: Comparing Prices Across Sources:

- **Input**: User selects a product and initiates a price comparison.
- **Expected Output**: Prices and deals for the selected product are compared across different sources, considering factors such as shipping costs, discounts, and availability.

5.4 Redirecting to Original Listings

TC-7: Seamless Redirection to External Website:

- **Input**: User clicks on a product listing to purchase.
- **Expected Output**: User is seamlessly redirected to the original listing on the external e-commerce website for purchase, ensuring a smooth transition without any interruptions.

6 Conclusion

This testing documentation outlines the approach, tools, and strategies for thoroughly testing our mobile application. By following this comprehensive testing plan, we can ensure the quality, reliability, and correctness of the application before deployment.