Testing and Quality Assurance Guide

Overview

The Testing and Quality Assurance Guide ensures the Weight Loss Coaching System operates seamlessly, meets functional requirements, and delivers an optimal user experience. This document outlines testing strategies, tools, and workflows for maintaining high-quality system performance.

Table of Contents

1. Testing Objectives

The primary goals of testing include:

- Validating system functionality and performance.
- Ensuring user data security and privacy.
- Identifying and resolving bugs or inconsistencies.
- Enhancing the system's scalability and reliability.

2. Types of Testing

a. Unit Testing

• Focuses on individual components (e.g., API endpoints, Bolt.new logic).

• Ensures each function behaves as expected.

• Example: Validate that /api/generate-plan returns a proper response based on input data.

b. Integration Testing

• Tests interactions between different modules (e.g., API and frontend).

• Ensures data flows seamlessly across components.

• Example: Verify that photo uploads trigger backend storage and are displayed on the dashboard.

c. End-to-End (E2E) Testing

• Simulates real user workflows to validate the entire system.

• Example: Test the onboarding process, from answering questions to receiving a personalized plan.

d. Performance Testing

• Measures system response times, scalability, and throughput.

• Example: Simulate 1,000 concurrent users uploading photos.

e. Security Testing

• Ensures data integrity, encryption, and access control.

• Example: Test for vulnerabilities in photo upload endpoints.

f. User Acceptance Testing (UAT)

• Involves end-users to validate usability and overall experience.

• Example: Gather feedback from beta testers on gamification features.

3. Test Scenarios and Cases

User Features

Scenario Expected Outcome

User uploads a photo Photo is stored securely, and progress comparison is updated.

User switches to Spanish during onboarding All text and labels update to Spanish in real-time.

User requests a new plan after feedback The system generates a revised plan within 24 hours.

Admin Features

Scenario Expected Outcome

Admin updates a supplement link The updated link is reflected across all associated user plans.

Admin accesses analytics Admin dashboard displays accurate engagement metrics and trends.

Distributor Features

Scenario Expected Outcome

Distributor customizes subdomain Distributor-branded subdomain is accessible to users.

Distributor views client supplement usage Client supplement engagement metrics are displayed correctly.

Gamification and Progress Tracking

Scenario Expected Outcome

User completes a daily streak User receives a badge, and leaderboard rankings are updated.

Weekly progress photos uploaded Side-by-side comparison is generated automatically.

4. Tools for Testing

Unit Testing Tools

• Jest:

• For testing API endpoints and backend logic.

• Example: Validate OpenAI integration with mocked API calls.

Integration Testing Tools

• Postman:

• For testing API interactions.

• Example: Verify /api/upload-photo processes valid images correctly.

• Cypress:

• For testing frontend-backend workflows.

• Example: Simulate a user navigating the dashboard.

End-to-End Testing Tools

• Selenium:

• Automates browser-based testing.

• Example: Test onboarding flows across different browsers.

• Playwright:

• Ensures multi-browser compatibility (e.g., Chrome, Firefox, Safari).

Performance Testing Tools

• JMeter:

• Simulates concurrent users to test system scalability.

• k6:

• Measures API response times under load.

Security Testing Tools

• OWASP ZAP:

• Identifies vulnerabilities like SQL injection or XSS.

• Burp Suite:

• Tests API endpoint security.

5. Continuous Integration and Deployment (CI/CD)

Setup

• Use platforms like GitHub Actions, CircleCI, or Travis CI for CI/CD pipelines.

• Automate tests during each code push or pull request.

Pipeline Example

1. Pre-Build Checks:

• Linting and static code analysis.

2. Build Stage:

• Compile and deploy backend and frontend components.

3. Test Stage:

• Run unit, integration, and E2E tests.

4. Deploy Stage:

• Deploy the application to staging or production environments.

Benefits

• Faster development cycles.

• Early detection of bugs.

6. Bug Tracking and Resolution

Bug Reporting Workflow

1. Identify and Log the Issue:

• Use a bug tracking tool (e.g., Jira, Trello, or GitHub Issues) to document:

• Steps to reproduce the bug.

• Screenshots or logs.

• Severity level (e.g., low, medium, high).

2. Assign Ownership:

• Assign the bug to the relevant developer or team.

3. Resolve and Test:

• Fix the bug and run regression tests to prevent reoccurrence.

4. Deploy Fix:

• Deploy hotfixes for critical issues or batch fixes in scheduled releases.

Common Bug Scenarios

Bug Cause Resolution

API response is delayed OpenAI API timeout Implement retry logic and fallback responses.

Photo comparison fails Unsupported file format or upload size Add pre-upload validation for images.

Streak reset incorrectly Data synchronization issue Check gamification logic and database triggers.

Testing Best Practices

1. Write Tests Early:

• Create unit tests alongside feature development.

2. Automate Repetitive Tests:

• Use automation tools for regression testing.

3. Prioritize Critical Features:

• Focus testing on user onboarding, progress tracking, and plan generation workflows.

4. Use Test Data:

• Create mock user profiles for testing without affecting production data.

This Testing and QA Guide ensures a robust framework for identifying, resolving, and preventing issues, leading to a seamless and reliable user experience.