

# **SUPPORTING NOTES – RECIPE MANAGER APPLICATION**

## **(FINAL SUBMISSION)**

### **1. Project Overview**

The Recipe Manager Web Application enables users to create, edit, view, delete, search, and filter recipes. Each recipe can contain a title, ingredients, detailed steps, and optional metadata such as category and preparation time. The system must handle structured text, multi-line content, and various input patterns while maintaining data integrity and usability.

This QA project evaluates the functional correctness, validation rules, formatting behavior, search logic, and user experience of the application across realistic and boundary input scenarios.

### **2. Project Scope**

In-Scope Functional Areas:

- Create Recipe
- Edit Recipe
- View Recipe
- Delete Recipe
- Search
- Filter
- Data Validation
- Negative & Edge Case flows
- Data Persistence

Out-of-Scope:

- Performance
- API
- Backend DB checks
- Full cross-browser testing
- Accessibility
- Localization

### **3. Project Objectives**

- Validate all recipe workflows
- Ensure correct validation and formatting
- Identify data-handling and UI issues
- Verify search, filter, sorting behaviors
- Confirm consistent CRUD functionality
- Test negative and boundary conditions
- Detect any security vulnerabilities

### **4. Test Data Summary**

Handwritten test data includes:

- Valid recipe content
- Multi-line ingredients
- Multi-line steps
- Optional metadata
- Invalid whitespace-only inputs
- Special characters
- Long text
- Script tags (<script>)

## **5. Supporting Notes – Test Plan**

The Test Plan defines the strategy, scope, approach, risk assessment, and assumptions. ISTQB-aligned testing was followed focusing on functional correctness, negative cases, formatting, and usability flows.

## **6. Supporting Notes – Test Scenarios**

Contains 18 scenarios covering happy path, validations, formatting, negative, search, filter, CRUD, and persistence flows.

Each scenario includes ID, title, objective, preconditions, priority, and type.

## **7. Supporting Notes – Test Cases**

Covers all functional and edge conditions with detailed fields including:

ID, Scenario ID, Objective, Preconditions, Steps, Test Data, Expected Result, Priority, Type.

Includes CRUD, validation, formatting, sorting, script handling, and negative test cases.

## **8. Supporting Notes – Bug Report**

Documents defects with ID, Title, Severity, Priority, Steps, Expected vs Actual, Evidence.

Covers functional bugs, formatting issues, validation failures, search defects, security issue (XSS), and data integrity issues.

## **9. Supporting Notes – Automation Proposal**

Recommended for automation:

- Validation cases
- Create/Edit
- Search/Filter
- Sorting

Not recommended:

- Visual layout
- Long text formatting

## **10. Final QA Summary**

The deliverables demonstrate strong functional coverage, well-structured scenarios, professional test cases, realistic data, and well-classified bugs.

Represents a complete and industry-standard QA submission.