Test Plan — Recipe Manager Web App

Test Item: Veg-Biryani
Author: Atul Hiray
Version: 1.0
Date: 21 Nov 2025

---

1. Purpose

This document defines the test plan for validating the Recipe Manager Web App using the candidate's Veg-Biryani (2 peoples) recipe as the primary test data. It identifies scope, out-of-scope, risks, assumptions, dependencies, test environment, tools, test data considerations and acceptance criteria.

---

2. Objectives

- Verify correctness, completeness and stability of CRUD operations for recipes.
- Validate input handling (format, units, special characters, line breaks).
- Verify image URL handling and media-related UX.
- Validate search, filter and data-validation behavior using the Veg-Biryani recipe variants.
- Identify and document risks and define test environment and tools.

---

3. Scope (In-scope)

Functional and non-functional testing areas covered for the Veg-Biryani recipe:

- **Create** recipe flow (all recipe fields: title, ingredients, quantities, units, steps, spices, images/URLs, tags, servings, cook time).
- **Read / View** recipe details page (display of ingredients, steps, quantities, media).
- **Edit / Update** recipe (modify quantities, steps, ingredients, image URL).

- **Delete** recipe and verify removal from lists and search.
- **Authentication/Authorization** flows affecting recipe creation/edit/delete (login, role-based restrictions).
- **Input validation** for fields (required fields, numeric units, allowed character sets, trimming, max lengths).
- **Image URL handling** (valid URL, broken URL, large image, unsupported format).
- **Search & Filter** (by title, ingredient e.g., “rice”, tag e.g., “vegetarian”, servings, spice level, preparation time).
- **Data persistence & consistency** across sessions and after edits.
- **Error states & messages** (validation errors, server errors, network timeouts).
- **Edge cases** driven by recipe data: line breaks in ingredients, unusual units, missing measurements, very large lists (multiple ingredients), special characters (e.g., “1/2 Teaspoon”, “2 Boul -washed”).

---

4. Out-of-Scope

- Performance/load testing beyond basic smoke (no large scale concurrency).
- Accessibility audit beyond basic checks (unless explicitly requested).
- Integration with external recipe marketplaces or 3rd-party payment systems.
- Mobile-native app testing (this plan targets the web app).
- Backend DB schema design verification (only behavior and persistence visible through UI/API).

---

5. Risks

- **Ambiguous ingredient formats** (e.g., “Rice 2 Boul -washed”, “Salt – 1/5 Tea spoon”) may cause parsing/display errors.
- **Image URL issues** (CORS, 404s) causing broken UI.

- **Authentication/Authorization misconfiguration** may allow unauthorized edits/deletes.
- **Inconsistent data validation** across create/edit endpoints (server vs client mismatch).
- **Internationalization / unit parsing** — users entering units in different formats may break numeric parsing.
- **Natural cooker-time wording** in steps may be misinterpreted by automated parsers (e.g., “wait until oil heated -7 min”).
- **Dependency outages** (CDN for images, authentication provider) may block testing.

---

6. Assumptions

- The Recipe Manager has standard CRUD UI and/or REST APIs for recipes.
- Authentication is available (login/logout) and roles (user/admin) exist; if no auth, tests for auth will be skipped.
- Test environment will have a separate test database that can be reset.
- The app accepts ingredient lists as either structured fields (name, quantity, unit) or as a free-text block; both variants will be tested.
- The app exposes error messages that are visible to testers or returned by APIs.

---

7. Dependencies

- Access to a test instance (URL), test user accounts (regular and admin), and test DB reset capability.
- Browsers for UI testing (Chrome latest, Firefox latest).
- Access to API endpoints (if automation uses APIs).
- Network access to host image URLs or ability to upload images.
- Test tools (listed below) installed and available.

---

8. Test Environment & Configuration

Environments:

- **DEV / QA**: Isolated test instance with test DB (recommended).
- **Staging**: Pre-production environment for final verification.

Hardware/Software:

- Desktop or laptop with modern OS (Windows/macOS/Linux).
- Browsers: Chrome (latest), Firefox (latest).
- Internet connectivity with stable bandwidth.

Services & Accounts:

- 2 test user accounts: test_user (regular), test_admin (admin).
- Image hosting: accessible static URLs for valid/invalid image tests.

Configuration / Data Reset:

- Mechanism to seed and wipe test DB before a test cycle.
- Time zone set to IST (Asia/Kolkata) if recipe times are localized.

---

9. Tools

- **Test management:** Excel/Google Sheets
- **Bug tracking:** Jira — templates with steps, severity, priority, attachments.
- **Automation:** Playwright (JS).
- **API testing:** Postman.
- **Data storage / sharing:** Google Drive / OneDrive for deliverables (Test Plan, Test Cases, Evidence).
- **Image hosting:** Static hosting (S3/Netlify) to serve test images and broken image URLs.

---

10. Test Data (Veg-Biryani recipe)

Primary record:

- **Title:** Veg-Biryani (2 peoples)
- **Servings:** 2
- **Ingredients:**
    - Cooker, cooking spoon (tools)
    - Rice — 2 Bowl — washed
    - Oil — 100 ml
    - Water — 500 ml
    - Salt — 1/5 teaspoon (note: ambiguous; treat as 0.2 tsp or test as invalid)
    - Jira (cumin) — 1 teaspoon
    - Elaichi (cardamom) — 2
    - Ginger-Garlic paste — 1/2 teaspoon
    - Peanuts — 50 gm
    - Cashew — 30 gm (10–12 pcs)
    - Curry leaves — 8–9 leaves
    - Curd — 2 teaspoon
    - Vegetables: Onion (1 medium, chopped), Carrot (1 medium), Potato (2 medium), Tomato (1 medium), Coriander (5–6 sticks chopped)
- **Spices:** Lal Mirch 1 tsp, Biryani masala 2 tsp, Garam masala 1/2 tsp
- **Procedure / Steps:** sequenced items with timings (gas on medium flame, oil heat 7 min, etc.) — treat as ordered steps.
- **Tags:** vegetarian, main course, 30–40 min (approx).
- **Image URL:** test with valid image, broken URL, large image, unsupported format.

Variations to be included:

- Positive: Well-formed structured ingredient entries (name, qty, unit).
- Negative: Ambiguous units (Boul, 1/5 Tea spoon), missing quantity, blank title, extremely long step text (>2000 chars).
- Edge: Ingredients with line breaks inside entries, duplicate ingredients, special characters, very large ingredient list (50 items), extremely large image (>5MB).

---

11. Test Types & Approach

- **Smoke / Sanity:** Basic create/view/edit/delete with canonical Veg-Biryani recipe.
- **Functional:** Full CRUD, search, filter, auth flows.
- **Input Validation:** Required fields, numeric parsing, unit validation, max lengths, prohibited characters.
- **Negative & Edge Cases:** Empty fields, malformed units, extremely long inputs, duplicate ingredients, line breaks in ingredient text.
- **Usability:** Readability of recipe page (quantities and step order).
- **Error Handling:** Server errors, network timeouts, 404 image handling, informative error messages.
- **Data Integrity:** Verify persistence after edits, rollback behavior on failed saves.
- **Security (basic):** Verify unauthorized users cannot edit/delete others' recipes.
- **Regression:** Re-run critical tests after changes.

---

12. Entry & Exit Criteria

Entry criteria:

- Test environment available and seeded.
- Test accounts provisioned.

- Access to the app (URL) and API keys (if required).
- Test data prepared (canonical and alternate variants).

Exit criteria:

- All high-severity defects found are fixed and retested.
- All planned test cases executed or documented as waived.
- Test summary report produced with pass/fail and outstanding risks.

---

13. Deliverables

- Approved Test Plan (this document).
- Test Scenarios (list/table).
- Test Case documents (≥25 detailed cases).
- Bug report list (mock and real defects).
- Automation proposal & framework outline.
- Test execution summary and evidence (screenshots / API logs).

---

14. Priority Areas & Recommendations (Risk-based)

1. **Input parsing for ingredients** — high priority due to many ambiguous entries (units, fractions, line breaks).
2. **Image URL handling** — medium-high priority to avoid broken UI.
3. **Authorization checks** — high priority to avoid data loss.
4. **Search / filter correctness** — medium priority to ensure discoverability (search by ingredient “rice”, “cashew”, tag “vegetarian”).
5. **Validation consistency (client vs server)** — medium priority.

---

15. Reporting & Metrics

- Track test execution (Pass/Fail/Blocked) — recommend spreadsheet or test tool.

- Defect metrics: count by severity, time to fix, reopen rate.
- Coverage: % of functional flows (CRUD, search, auth) executed.
- Risk status: open/mitigated.

---

16. Timeline

- **Day 0:** Setup environment, seed test data.
- **Day 1:** Smoke tests + create canonical Veg-Biryani.
- **Day 2–3:** Functional tests + input validation cases.
- **Day 4:** Negative/edge tests (line breaks, malformed units).
- **Day 5:** Search/filter & auth tests.
- **Day 6:** Error state and image URL tests.
- **Day 7:** Regression, bug verification, produce final report.

(Adjust to actual 10-hour assessment split — prioritize smoke + critical functional + input validation.)

---

17. Approval

- **Prepared by:** Atul Hiray
- **Reviewed / Approved by:** [QA Lead]