Automation Proposal - Recipe Manager Web App

TC ID	Test Case Name	Description	Test Data	Why Automate
TC_A01	Create Recipe	Verify user can create a full recipe with required fields, image URL, ingredients & steps.	**Title:** Veg-Biryani (2 peoples) **Ingredients:** Rice 2 Bowl, Oil 100 ml, Water 500 ml, Salt 0.2 tsp, Jeera 1 tsp, Elaichi 2, GingerGarlic 0.5 tsp, Peanuts 50 gm, Cashew 30 gm, Curry Leaves 8–9, Curd 2 tsp, Vegetables list etc. **Steps:** Full procedure lines **Tags:** Main Course, Vegetarian **Image URL:** Valid URL	High business value, core flow, required in every release, stable & deterministic.
TC_A02	Edit Recipe	Ensure editing updates ingredient quantity & steps correctly and persists after save.	Update Oil from 100 ml → 120 ml Add one additional cooking step Verify UI shows updated data & backend/API consistency	Common regression area, repeatable, low flakiness.
TC_A03	Input Validation	Confirm UI rejects invalid image URLs & invalid numeric values.	Invalid URL: "not_a_url" Invalid Qty: Water = -100 or "five hundred" Ensure proper error messages	Negative tests are deterministic and reliable for automation.
TC_A04	Search & Filter	Validate search & filter return correct recipe.	Search: “biryani” Filter: Vegetarian, Ingredient = Rice, Servings = 2 peoples	Core functionality, logic-heavy, prone to regressions, predictable output.
TC_A05	Delete Recipe	Verify deletion works and recipe is removed from list + search results.	Delete Veg-Biryani recipe Confirm recipe disappears from UI If soft-delete: verify it appears in Trash & can be restored	Safety-critical flow, strong candidate for regression automation.

Justification

- These flows are deterministic, fast, and executed frequently as regression checks. They verify core CRUD and content rendering which are high-risk for business.
- Automating create/edit/delete ensures test data lifecycle: create recipe, verify, edit, verify, delete (teardown).
- Image URL tests validate front-end rendering and fallback logic programmatically.

Pseudo-automation framework (Playwright recommended)

- **Language:** Playwright (Node.js). I prefer Playwright for faster test runs and built-in waiting.
- **Test runner:** Playwright Test (or Jest + Playwright).
- **Assertions/reporting:** Playwright Test + Allure or built-in reporter.

Folder Structure

pages > struct.txt

```
/automation
│
├── package.json
├── playwright.config.js
│
├── tests/
│   ├── createRecipe.spec.js       // TC_A01
│   ├── editRecipe.spec.js         // TC_A02
│   ├── inputValidation.spec.js    // TC_A03
│   ├── searchFilter.spec.js       // TC_A04
│   └── deleteRecipe.spec.js       // TC_A05
│
├── pages/
│   ├── BasePage.js
│   ├── RecipeListPage.js
│   ├── RecipeCreatePage.js
│   ├── RecipeDetailPage.js
│   └── LoginPage.js               // optional
│
├── data/
│   ├── veg_biryani.json
│   ├── invalid_urls.json
│   └── negative_cases.json
│
├── utils/
│   ├── helpers.js
│   └── apiClient.js               // optional for backend verification
│
└── ci/
    └── github-actions.yml
```

Sample Page Object (Playwright + JavaScript)

```
const { test, expect } = require("@playwright/test");
const { RecipeCreatePage } = require("../pages/RecipeCreatePage");
const data = require("../data/veg_biryani.json");

test("TC_A01 - Create Recipe", async ({ page }) => {
 const createPage = new RecipeCreatePage(page);

 await page.goto("/recipes/new");

 await createPage.enterTitle(data.title);

 for (const ing of data.ingredients) {
  await createPage.addIngredient(ing.name, ing.qty, ing.unit);
 }

 await createPage.setSteps(data.steps);
 await createPage.setImageUrl(data.image_url);

 await createPage.save();

 // Assertions
 await expect(page.locator("h1.recipe-title")).toHaveText(data.title);
});
```

Sample Test File (Playwright + JavaScript)

```
const { test, expect } = require("@playwright/test");
const { RecipeCreatePage } = require("../pages/RecipeCreatePage");
const data = require("../data/veg_biryani.json");

test("TC_A01 - Create Recipe", async ({ page }) => {
 const createPage = new RecipeCreatePage(page);

 await page.goto("/recipes/new");

 await createPage.enterTitle(data.title);

 for (const ing of data.ingredients) {
  await createPage.addIngredient(ing.name, ing.qty, ing.unit);
 }

 await createPage.setSteps(data.steps);
 await createPage.setImageUrl(data.image_url);

 await createPage.save();

 // Assertions
 await expect(page.locator("h1.recipe-title")).toHaveText(data.title);
});
```