Here's a formatted version of your content, optimized for Confluence:

API Automation Framework Using Playwright and Cucumber.js

🧭 1. Overview

Objective

This document outlines an AI-assisted API Automation Framework built using Playwright and Cucumber.js. It automates API endpoint validation directly from Swagger/OpenAPI specifications, with comprehensive schema validation and visual reporting.

Key Highlights

- **Config-driven approach**: Read API specification and authentication from `.env` file.
- **Auto-generates Playwright + Cucumber test suites** from Swagger/OpenAPI (JSON or URL).
- **Schema validation**: Validates response structure using Ajv (required fields + data types).
- **Allure reporting**: Interactive HTML dashboards with test statistics and trends.
- **Automatic retry logic**: 3 attempts per scenario for transient failures.
- **Flexible authentication**: Supports Basic Auth, Bearer Token, or none.
- **Seamlessly integrates with GitHub Copilot**.

Key Technologies

- 🧩 **Playwright** – `APIRequestContext` for HTTP calls and response handling.
- 🧠 **Cucumber.js v10+** – BDD test framework with retry configuration.

- ✅ **Chai v5+** – Assertion library for response validation.
- 🔒 **Ajv v8+** – JSON Schema validation for API contract compliance.
- 📊 **Allure** – Primary reporting tool with interactive dashboards.
- 🤖 **GitHub Copilot** – AI-powered test generation from `.copilot-instructions`.
- 🔐 **dotenv** – Environment variable management.

Purpose

To create a configuration-driven, schema-validated, and visually reportable API automation suite that:

- Runs against Swagger/OpenAPI-defined endpoints.
- Validates API contract compliance (schema validation).
- Categorizes failures (Critical, Validation Gaps, Schema Violations).
- Generates multiple report formats (Cucumber HTML/JSON, Allure dashboard, `TEST_SUMMARY.md`).
- Requires zero manual intervention after initial `.env` configuration.

⚙️ 2. Prerequisites

Required Software

- Node.js v18+ and npm
- Active GitHub Copilot Business License
- Playwright v1.45+ (APIRequestContext support)
- Cucumber.js v10+ (BDD framework with retry)

- Swagger/OpenAPI specification (JSON file or URL)

Required Libraries

- **Chai v5+** – Assertions
- **Ajv v8+ & ajv-formats** – JSON Schema validation
- **Allure (allure-cucumberjs, allure-commandline)** – Interactive reporting
- **dotenv** – Environment configuration

💻 Supported IDEs

IDE	Status	Notes
VS Code	✅ Recommended	Seamless Copilot & GHCP integration
IntelliJ IDEA	✅ Supported	Minor confirmation prompts during auto-fix

---

🧩 3. Architecture & Process Flow

High-Level Flow

1. **Configuration-Driven**: All settings in `.env` (no hardcoded values).
2. **Schema Validation**: Validates response structure against API specification.
3. **Automatic Retry**: Cucumber retries failed scenarios 3 times.
4. **Failure Categorization**: Critical, Validation Gaps, Schema Violations, Not Applicable.
5. **Multiple Reports**: Cucumber, Allure, and markdown summaries.

🚀 4. Setup & Configuration

Step 1: Configure `.env` File

Create a `.env` file at the project root.

Step 2: Generate Tests Using Copilot

Simply provide the following prompt to GitHub Copilot.

Step 3: Execute Tests

Run your tests using Cucumber and Playwright.

📂 5. Project Structure

🧪 6. Test Coverage

For each endpoint operation:

1. Positive Scenarios (200/201/204)

- Validates successful operations per Swagger/OpenAPI definitions.
- **Schema validation** performed on responses:
    - `GET (200)`: Always validate response schema.
    - `POST/PUT (200/201)`: Validate if response returns an object.
- **If test fails** → indicates API bug (not test bug).

2. Negative Scenarios

HTTP Method	Tested Negative Codes
`GET`	401, 404, 500
`POST`	400, 401, 404, 500
`PUT`	400, 404, 500

`DELETE`	400, 404, 500

Note: Negative tests may fail if:

- API lacks input validation (returns 200 instead of 400) → Validation Gap.
- API returns the wrong error code (returns 500 instead of 400) → Wrong Error Code.

These are documented in `TEST_SUMMARY.md` as expected failures.

🔄 7. Retry & Failure Handling

Automatic Retry Logic

- **Cucumber configuration**: `retry: 3` (3 attempts per scenario).
- **Purpose**: Handle transient environment issues (network glitches, temporary service issues).
- **Behavior**:
    - Test fails → automatically retries up to 3 times.
    - Passes on retry → transient issue (test passes).
    - Fails all 3 retries → genuine API issue (marked as **FAILED**).

Failure Categorization

All failures are documented in `TEST_SUMMARY.md`:

- **Positive Scenario Failures (CRITICAL)**: Swagger-defined success responses that fail.
    - Example: `POST` expects `200` but returns `500`.
    - **Action**: Indicates API bugs.

- **Negative Scenario Failures - Validation Gaps**: API returns `200` instead of `400` (missing input validation).
    - **Documented in**: "Not Applicable Test Cases".
- **Negative Scenario Failures - Wrong Error Code**: API returns `500` instead of `400/404`.
    - **Documented in**: "Failed Test Cases → Negative Scenarios".
- **Schema Validation Failures (CRITICAL)**: Response structure doesn't match API specification.
    - Examples: Missing required fields, wrong data types, unexpected fields.
    - **Action**: API contract violation – requires immediate attention.

Parallel Execution Support

The framework is configured for parallel execution via Cucumber.js (currently set to `parallel: 1` for stability).

📊 8. Reporting

Three report types are generated:

1. Cucumber HTML/JSON Reports

- **Location**: `cucumber-report.html`, `cucumber-report.json`.
- **Content**: Basic test execution summary with step-level details.

2. Allure Report (Primary)

- **Location**: `index.html`.
- **Content**: Interactive dashboard with:

    - Test statistics and trends
    - Categorized failures
    - Step-level logs and stack traces
    - Drill-down capabilities
- **Open with**: `npm run allure:open`.

3. `TEST_SUMMARY.md`

- **Content**:
    - Test Results (Total, Passed, Failed, Success Rate).
    - Test Coverage by Endpoint (✅/❌/⚠️).
    - Failed Test Cases (categorized by type).
    - Not Applicable Test Cases (validation gaps).
    - API Issues Requiring Attention (Critical vs Behavior Issues).
    - Next Steps (prioritized action items).

🧪 9. Sample Test Case

- Each test is self-contained with automatic retry (3 attempts).
- Schema validation is performed on successful responses.
- Failures are categorized and documented in `TEST_SUMMARY.md`.

🔍 10. Key Observations

- **Configuration-Driven Approach**: All configuration centralized in `.env` file. No hardcoded values in test code.
- **Schema Validation**: Catches response structure issues early. Documented as **CRITICAL** failures (API contract violations).
- **Authentication Flexibility**: Supports multiple auth types: basic, bearer, none (configured via `AUTH_TYPE` in `.env`).
- **IDE Integration**:
    - **VS Code**: Seamless Copilot integration, recommended.
    - **IntelliJ IDEA**: Fully supported with minor confirmation prompts.
- **Test Philosophy**: Tests NEVER modified to match API behavior. Failures indicate API issues, not test issues.

🏁 11. Conclusion

This framework demonstrates how Playwright + Cucumber + Ajv + Allure + GitHub Copilot can automate comprehensive API testing using Swagger/OpenAPI specifications with:

- ✅ Zero manual intervention after `.env` configuration.
- ✅ Schema validation for API contract compliance.
- ✅ Flexible authentication (basic/bearer/none).
- ✅ Automatic retry logic (3 attempts per scenario).
- ✅ Visual reporting with Allure dashboards.
- ✅ Failure categorization (Critical, Validation Gaps, Schema Violations).
- ✅ AI-assisted test generation via GitHub Copilot.

Key Differences from Previous Version:

- ✅ Added `.env` configuration-driven approach.
- ✅ Added schema validation with Ajv.
- ✅ Added Allure reporting as primary reporter.
- ✅ Added authentication flexibility (basic/bearer/none).
- ✅ Updated failure categorization (4 categories including schema violations).
- ✅ Clarified retry logic (Cucumber scenario-level, 3 attempts).
- ✅ Added detailed reporting section (3 report types).
- ✅ Updated tech stack versions (Cucumber v10+, Chai v5+, Ajv v8+).
- ✅ Removed "self-healing" terminology (it's retry logic, not self-healing).
- ✅ Added complete project structure with comments.

This version is now ready to be included in your Confluence documentation!