General API Automation Strategy (Postman-Based Principles)

Foundation: Formalizing successful test patterns observed in the E-commerce API Postman collection (GET, POST, Negative). **Objective:** To establish a scalable, language-agnostic framework that guarantees **functional behavior** and **data contract integrity** across all public API services.

1. Core Testing Mandate

The automation scope covers the three fundamental test types demonstrated in the collection:

1. **Functional Success (GET/Retrieve):** Verify successful data retrieval (`200 OK`) and that response arrays are populated and not empty.
2. **Resource Creation (POST/Cart):** Validate that resource creation results in an expected status code (`200`/`201 Created`) and returns the **unique identifier** (e.g., `id`) of the new resource.
3. **Negative/Error Handling (Invalid ID):** Ensure all invalid or edge-case requests return the correct **failure status codes** (`400`, `404`) and that the response payload is empty or minimal (graceful failure).

2. Technical Strategy: Service Object Model (SOM)

The framework must use a **Service Object Model (SOM)** to separate test execution from API interaction, enhancing maintainability:

- **Test Layer:** Contains test logic and assertions (e.g., "expect product list to be non-empty").
- **Service Layer:** Contains methods for making requests (e.g., `api.getProducts()`) and handles URL, headers, and payload construction.
- **Contract Layer:** Centralizes **JSON Schema definitions** for every critical endpoint. This formalizes the Postman field validation (e.g., `pm.expect(jsonData[0]).to.have.property('id')`) into strict, self-documenting schema checks.

3. Tool Stack Recommendation

A modern stack is recommended to replace raw Postman scripting with reusable code:

Component	Tool Function	Rationale
Language	TypeScript/JavaScript	Familiarity with Postman's underlying JS testing environment.
HTTP Client	**Axios**	Efficient, promise-based request handling.
Test Runner	**Jest / Mocha**	Fast execution and clear assertion syntax.
Data Validation	**Ajv**	Enforces the strict data contracts defined in the Contract Layer.