### **API Test cases documentation.**

**Technologies used:** Playwright and Typescript

**Goal:** Cover the endpoints possibilities with the test cases below and the information provided in the swagger documentation.

This documentation is revised by LLM (ChatGPT).

### **Endpoint: GET /api/MobileFoodTrucks/searchByName**

**Purpose:** Search food trucks by name (optionally filtered by status).

**Test Cases:**

1. **should return 400 when name is missing** Request sent without name parameter.  
    **Expect:** 400 Bad Request.
2. **should return 200 and list when name matches** Query with name=Taco.  
    **Expect:** 200 OK, array of results containing at least one item. At least one item's applicant should include the word "Taco" (case-insensitive).
3. **should return only APPROVED trucks when name and status=APPROVED** Query with name=Taco&status=APPROVED.  
    **Expect:** 200 OK, all returned items must have status equal to "APPROVED".
4. **should return only REQUESTED trucks when name and status=REQUESTED** Query with name=Taco&status=REQUESTED.  
    **Expect:** 200 OK, all returned items must have status equal to "REQUESTED".
5. **should return only EXPIRED trucks when name and status=EXPIRED** Query with name=Taco&status=EXPIRED.  
    **Expect:** 200 OK, all returned items must have status equal to "EXPIRED".
6. **should return only SUSPENDED trucks when name and status=SUSPENDED** Query with name=Taco&status=SUSPENDED.  
    **Expect:** 200 OK, all returned items must have status equal to "SUSPENDED".
7. **should return empty array when no matches found** Query with name=NoSuchTruck.  
    **Expect:** 200 OK, empty array.

### **Endpoint: GET /api/MobileFoodTrucks/searchByStreet**

**Purpose:** Search food trucks by street name.

**Test Cases:**

1. **should return 200 and trucks for a valid street** Query with street=A St.  
    **Expect:** 200 OK, array with at least one item. Each truck’s address should contain "A St" (case-insensitive).
2. **should return empty array when street has no trucks** Query with street=Nowhere.  
    **Expect:** 200 OK, empty array.
3. **should return 400 if street param is missing** Request without street query parameter.  
    **Expect:** 400 Bad Request.

### **Endpoint: GET /api/MobileFoodTrucks/nearestFoodTrucks**

**Purpose:** Retrieve the nearest food trucks by geographic coordinates.

**Test Cases:**

1. **should return up to 5 closest APPROVED trucks by default** Query with latitude=37.8&longitude=-122.4 and no status specified.  
    **Expect:** 200 OK, array of up to 5 trucks. All must have status = "APPROVED".
2. **should return REQUESTED trucks when status=REQUESTED** Query with latitude=37.8&longitude=-122.4&status=REQUESTED.  
    **Expect:** 200 OK, all returned items must have status = "REQUESTED".
3. **should return 400 if latitude is missing** Query with only longitude=-122.4.  
    **Expect:** 400 Bad Request.
4. **should return 400 if longitude is missing** Query with only latitude=37.8.  
    **Expect:** 400 Bad Request.
5. **should return empty array if no trucks match status** Query with latitude=37.8&longitude=-122.4&status=NONEXISTENT.  
    **Expect:** 200 OK, empty array.
6. **should return 400 with validation errors for invalid latitude and longitude**Query with latitude=AAAA and longitude=-AAAA and status=APPROVED.  
   **Expect**: 400 Bad Request. Response should include validation error messages for both latitude and longitude.
7. **should return 200 and an empty array for far west coordinates (Pacific Ocean)** Query with latitude=0&longitude=-150.  
    **Expect:** 200 OK, empty array.
8. **should return 200 and an empty array for far north coordinates (Alaska)** Query with latitude=65&longitude=-150.  
    **Expect:** 200 OK, empty array.
9. **should return 200 and an empty array for east-coast coordinates (New York)** Query with latitude=40.7128&longitude=-74.006.  
    **Expect:** 200 OK, empty array.

## **Bugs**

**1. Silent acceptance of invalid status values** status=NONEXISTENT or status=INVALID returns 200 OK with an empty array.  
 This silently ignores invalid input instead of rejecting it.  
 **Expected behavior:** API should return 400 Bad Request for unsupported status values.

**2. No backend validation for coordinates out of bounds** The API returns 200 OK even for coordinates far outside San Francisco (e.g., Alaska, 0/0, Pacific Ocean).  
 **Expected behavior:** Backend should validate latitude/longitude and return 400 for out-of-bound values.

**3. Inconsistent response when no matches are found** Returning a 200 OK with an empty array is fine, but it's not consistent across use cases.  
 **Improvement:** Consider returning a message in the body when the array is empty, e.g., { "message": "No results found" }.

**4. Status filtering is case-sensitive**In searchByName endpoint, status being case-sensitive may break user expectations.  
**Expected behavior:** API should normalize or reject incorrect casing with a clear message.

**5. No validation for excessive input length** Inputs like name= with very long strings (e.g. 1000 characters) are not validated or limited.  
 **Risk:** May lead to performance issues or injection attempts.

**6. No response caching or pagination** Endpoints like /searchByName may return large result sets.  
 **Improvement:** Consider paginating or limiting results to avoid overloading responses.

## **Improvements**

**1. Add limit query parameter to /nearestFoodTrucks** By default, the API returns up to 5 trucks. Allowing a custom limit (e.g., limit=10) would make the API more flexible.

**2. Improve error responses** For every 400 Bad Request, include a meaningful, standardized error message.  
 This makes debugging and integration much easier.

**3. Document query behavior and constraints** There’s no information in the API output indicating what values are valid for status, or the coordinate limits for SF.  
 Consider adding this in the API docs or as a metadata endpoint.

**4. Reject unsupported query parameters** Currently, parameters like ?foo=bar doesn't trigger validation errors. Add logic to reject or ignore unsupported fields explicitly.

**5. Improve API documentation**The api documentation doesn’t provide information on which fields are required in each endpoint. This should be informed as a label in the Swagger fields