**Test Cases**

**Test Case 1: Verify Unique Customer Count by Contract ID**

* **Test ID**: TC001
* **Objective**: Validate that the application returns the correct count of unique customer IDs per contract ID.
* **Preconditions**: Application is running, and sample data is loaded.
* **Test Steps**:
  1. Send a GET request to /api/report/uniqueCustomersByContract.
* **Expected Result**: The response should contain the unique customer count per contract ID as per the sample data:
  1. Contract ID 2345: 3 unique customers
  2. Contract ID 2346: 2 unique customers

**Test Case 2: Verify Unique Customer Count by Geozone**

* **Test ID**: TC002
* **Objective**: Verify that the application returns the correct count of unique customer IDs per geozone.
* **Preconditions**: Application is running, and sample data is loaded.
* **Test Steps**:
  1. Send a GET request to /api/report/uniqueCustomersByGeozone.
* **Expected Result**: The response should contain the unique customer count per geozone:
  1. Geozone us\_east: 1 unique customer
  2. Geozone us\_west: 2 unique customers
  3. Geozone eu\_west: 2 unique customers

**Test Case 3: Verify Average Build Duration by Geozone**

* **Test ID**: TC003
* **Objective**: Ensure that the application correctly calculates and returns the average build duration for each geozone.
* **Preconditions**: Application is running, and sample data is loaded.
* **Test Steps**:
  1. Send a GET request to /api/report/averageDurationByGeozone.
* **Expected Result**: The response should return the average build duration for each geozone:
  1. Geozone us\_east: 3445.0 seconds
  2. Geozone us\_west: 2216.0 seconds
  3. Geozone eu\_west: 4222.0 seconds

**Test Case 4: Verify Unique Customer List by Geozone**

* **Test ID**: TC004
* **Objective**: Ensure that the application provides a unique list of customer IDs for each geozone.
* **Preconditions**: Application is running, and sample data is loaded.
* **Test Steps**:
  1. Send a GET request to /api/report/customerListByGeozone.
* **Expected Result**: The response should contain a unique list of customer IDs for each geozone:
  1. Geozone us\_east: [2343225]
  2. Geozone us\_west: [1223456, 1233456]
  3. Geozone eu\_west: [3244332, 3244132]

**Test Case 5: Verify Response Time for Large Data Input**

* **Test ID**: TC005
* **Objective**: Ensure acceptable response time (< 1 second) with large data inputs.
* **Preconditions**: Application is running with large data input (thousands of entries).
* **Test Steps**:
  1. Send requests to each endpoint (uniqueCustomersByContract, uniqueCustomersByGeozone, etc.).
* **Expected Result**: Each request should respond within 1 second.

**Test Case 6: Verify Error Handling for Invalid Data**

* **Test ID**: TC006
* **Objective**: Validate that the application handles invalid data gracefully.
* **Preconditions**: Application is running.
* **Test Steps**:
  1. Modify the input data format (e.g., remove a field or add an invalid character).
  2. Send a request to /api/report/uniqueCustomersByContract.
* **Expected Result**: The application should respond with an error message or HTTP 400 Bad Request.

**Test Case 7: Boundary Testing - Minimum and Maximum Build Durations**

* **Test ID**: TC007
* **Objective**: Test the application with minimum (e.g., 0s) and maximum (e.g., 9999s) build durations.
* **Preconditions**: Application is running with modified input data.
* **Test Steps**:
  1. Update build durations to 0s and 9999s.
  2. Send a request to /api/report/averageDurationByGeozone.
* **Expected Result**: The response should correctly calculate the average, including 0s and 9999s.

**Test Case 8: Verify Application Stability on Rapid Requests**

* **Test ID**: TC008
* **Objective**: Test application stability by sending multiple requests to each endpoint in quick succession.
* **Preconditions**: Application is running.
* **Test Steps**:
  1. Send 100 rapid requests to /api/report/uniqueCustomersByContract.
  2. Repeat for each endpoint.
* **Expected Result**: The application should handle all requests without crashes or errors.

**Test Case 9: Verify Data Consistency Across Requests**

* **Test ID**: TC009
* **Objective**: Ensure data consistency across multiple requests with no changes in data between requests.
* **Preconditions**: Application is running.
* **Test Steps**:
  1. Send multiple requests to /api/report/uniqueCustomersByContract.
  2. Compare results for consistency.
* **Expected Result**: The same output should be returned across all requests, with no data inconsistency.