

Test Results Report

This report summarizes the execution of test cases for the PizzaDronz system. The testing process aims to verify the correctness, robustness, and efficiency of different system components, including API endpoints, order validation, pathfinding algorithms, and combinatorial logic.

Summary of test cases executed

Test files can be found in

<https://github.com/ediithhh/SoftwareTesting/tree/main/src/test/java/uk/ac/ed/inf>

- Total tests executed: 147
- Tests passed: 147

Test Category	Number of Tests	Passed tests
Unit Tests	40	40/40
Combinatorial Tests	20	20/20
Model-Based Tests	32	32/32
Structural Tests	13	13/13
API Tests	42	42/42

- Test completion time: 11sec 695ms

```
✓ Tests passed: 148 of 148 tests – 11 sec 695 ms
✓ <default package> 11sec 695 ms
  ✓ ModelBasedTests 2 sec 5 ms
    ✓ APITests 9sec 64 ms
      ✓ StructuralTests 28 ms
      ✓ OrderValidationImplTestModelBased 133 ms
      ✓ NoFlyZoneServiceTest 20 ms
      ✓ PathfindingAlgorithmTest 31 ms
      ✓ RestaurantServiceTest 9 ms
      ✓ PizzaDronzApplicationTests 2 ms
      ✓ PathfindingAlgorithmTestsCombinatorial 335 ms
      ✓ PathfindingAlgorithmTestModelBased 33 ms
      ✓ CombinatorialTests 23 ms
      ✓ OrderValidationImplTestUnit 7 ms
      ✓ CentralAreaServiceTest 5 ms
      2025-01-23T22:14:38.388Z INFO 14364 --- [PizzaDronz] [
      2025-01-23T22:14:38.393Z INFO 14364 --- [PizzaDronz] [
      2025-01-23T22:14:38.396Z INFO 14364 --- [PizzaDronz] [
        "orderNo": "77DAD717",
        "pizzasInOrder": [
          { "name": "R3: Super Cheese", "priceInPence":
        ]
      }
      2025-01-23T22:14:38.400Z INFO 14364 --- [PizzaDronz] [
      2025-01-23T22:14:38.402Z INFO 14364 --- [PizzaDronz] [
        "position1": { "lng": -3.192473, "lat": 55.946233
        "position3": { "lng": -3.192473, "lat": 55.946233
      }
```

Examples of performance tests:

```

@Test
void testCalculatePathPerformance() {
    String url = "https://localhost:8080/api/v1/calculatePath";
    String requestBody = """
        {
            "start": "2025-01-23T22:14:38.388Z",
            "end": "2025-01-23T22:14:38.393Z",
            "order": "77DAD717",
            "pizzasInOrder": [
                { "name": "R3: Super Cheese", "priceInPence": 1000 }
            ],
            "position1": { "lng": -3.192473, "lat": 55.946233 },
            "position3": { "lng": -3.192473, "lat": 55.946233 }
        }
    """;
    long startTime = System.nanoTime();
    Response response = sendHttpRequest(url, requestBody, Order.class);
    long endTime = System.nanoTime();
    long duration = endTime - startTime;
    logger.info("Execution time for calculatePath: {} ms", duration);
    assertEquals("Execution time for calculatePath is less than 100ms", duration, isLessThan(100_000_000L));
    assertEquals("Response status is 200 OK", response.getStatusCode(), is(200));
}

@Test
void testCalculatePathPerformance() {
    String url = "https://localhost:8080/api/v1/calculatePath";
    String requestBody = """
        {
            "start": "2025-01-23T22:14:38.388Z",
            "end": "2025-01-23T22:14:38.393Z",
            "order": "77DAD717",
            "pizzasInOrder": [
                { "name": "R3: Super Cheese", "priceInPence": 1000 }
            ],
            "position1": { "lng": -3.192473, "lat": 55.946233 },
            "position3": { "lng": -3.192473, "lat": 55.946233 }
        }
    """;
    long startTime = System.nanoTime();
    Response response = sendHttpRequest(url, requestBody, Order.class);
    long endTime = System.nanoTime();
    long duration = endTime - startTime;
    logger.info("Execution time for calculatePath: {} ms", duration);
    assertEquals("Execution time for calculatePath is less than 100ms", duration, isLessThan(100_000_000L));
    assertEquals("Response status is 200 OK", response.getStatusCode(), is(200));
}

@Test
void testCalculatePathPerformance() {
    String url = "https://localhost:8080/api/v1/calculatePath";
    String requestBody = """
        {
            "start": "2025-01-23T22:14:38.388Z",
            "end": "2025-01-23T22:14:38.393Z",
            "order": "77DAD717",
            "pizzasInOrder": [
                { "name": "R3: Super Cheese", "priceInPence": 1000 }
            ],
            "position1": { "lng": -3.192473, "lat": 55.946233 },
            "position3": { "lng": -3.192473, "lat": 55.946233 }
        }
    """;
    long startTime = System.nanoTime();
    Response response = sendHttpRequest(url, requestBody, Order.class);
    long endTime = System.nanoTime();
    long duration = endTime - startTime;
    logger.info("Execution time for calculatePath: {} ms", duration);
    assertEquals("Execution time for calculatePath is less than 100ms", duration, isLessThan(100_000_000L));
    assertEquals("Response status is 200 OK", response.getStatusCode(), is(200));
}

@Test
void testCalculatePathPerformance() {
    String url = "https://localhost:8080/api/v1/calculatePath";
    String requestBody = """
        {
            "start": "2025-01-23T22:14:38.388Z",
            "end": "2025-01-23T22:14:38.393Z",
            "order": "77DAD717",
            "pizzasInOrder": [
                { "name": "R3: Super Cheese", "priceInPence": 1000 }
            ],
            "position1": { "lng": -3.192473, "lat": 55.946233 },
            "position3": { "lng": -3.192473, "lat": 55.946233 }
        }
    """;
    long startTime = System.nanoTime();
    Response response = sendHttpRequest(url, requestBody, Order.class);
    long endTime = System.nanoTime();
    long duration = endTime - startTime;
    logger.info("Execution time for calculatePath: {} ms", duration);
    assertEquals("Execution time for calculatePath is less than 100ms", duration, isLessThan(100_000_000L));
    assertEquals("Response status is 200 OK", response.getStatusCode(), is(200));
}
```

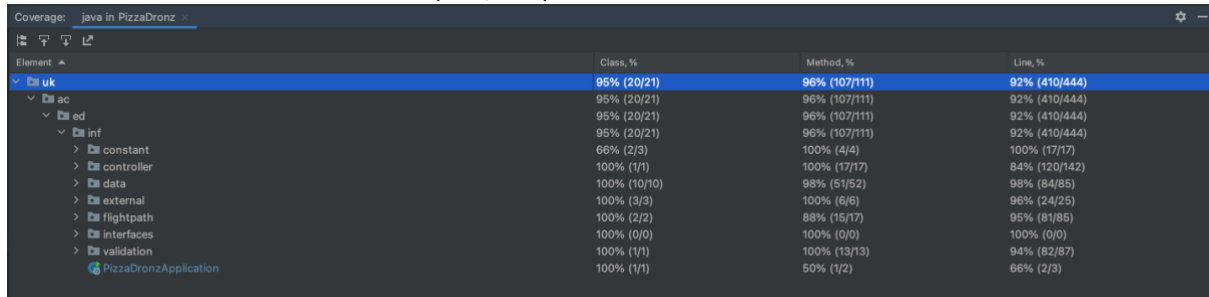
Test log for execution time of calcDeliveryPath:

```
INFO 25229 --- [PizzaDronz] [http-nio-auto-1-exec-1] o.a.c.c.C.[Tomcat].[localhost].[] : Initializing Spring DispatcherServlet 'dispatcherServlet'
INFO 25229 --- [PizzaDronz] [http-nio-auto-1-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
INFO 25229 --- [PizzaDronz] [http-nio-auto-1-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 0 ms
INFO 25229 --- [PizzaDronz] [main] uk.ac.ed.inf.APITests : Response from http://localhost:55403/calcDeliveryPath: Status 200 OK, Body [Lng
INFO 25229 --- [PizzaDronz] [main] uk.ac.ed.inf.APITests : Execution time for calcDeliveryPath: 463 ms
```

Test logs can be found in <https://github.com/ediithhh/SoftwareTesting/blob/main/logs/test.log>

Code Coverage Report

- Overall code coverage:
 - Classes covered: 95% (20/21)
 - Methods covered: 96% (107/111)
 - Lines covered: 92% (410/444)



The screenshot shows the 'Coverage' tool window in IntelliJ IDEA, displaying a hierarchical view of code coverage for the 'PizzaDronz' project. The table lists elements from the 'uk' package down to the 'PizzaDronzApplication' class, showing class, method, and line coverage percentages and counts.

Element	Class, %	Method, %	Line, %
uk	95% (20/21)	96% (107/111)	92% (410/444)
ac	95% (20/21)	96% (107/111)	92% (410/444)
ed	95% (20/21)	96% (107/111)	92% (410/444)
inf	95% (20/21)	96% (107/111)	92% (410/444)
constant	66% (2/3)	100% (4/4)	100% (17/17)
controller	100% (1/1)	100% (17/17)	84% (120/142)
data	100% (10/10)	98% (51/52)	98% (84/85)
external	100% (3/3)	100% (6/6)	96% (24/25)
flightpath	100% (2/2)	88% (15/17)	95% (81/85)
interfaces	100% (0/0)	100% (0/0)	100% (0/0)
validation	100% (1/1)	100% (13/13)	94% (82/87)
PizzaDronzApplication	100% (1/1)	50% (1/2)	66% (2/3)