# TEST DESCRIPTION

**TEST NAME OR ID**: T1-T4

**Test Type**: Integration Testing

**Description**: Test the internal logic of the validateShipment() and checking the capacity of the truck afterward.

**Setup:** VS code integration test template, functions, and a main() to execute the test function.

**Test Function**:

validateShipment()

isTruckCapacitySufficient()

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Test data for T1 (Valid shipment and sufficient truck capacity) | Truck truck = { 1, 1000.0, 10.0, 0, 0, {'B', {}}, nullptr }; PackageInfo shipment = { 800.0, 5.0, {10, 10} }; | 1 | 1 | PASS |
| Test data for T2 (Valid shipment but insufficient truck capacity) | Truck truck = { 2, 500.0, 5.0, 0, 0, {'Y', {}}, nullptr }; PackageInfo shipment = { 600.0, 3.0, {1, 1} }; | 0 | 0 | PASS |
| Test data for T3 (Invalid shipment: zero weight) | Truck truck = { 3, 2500.0, 100.0, 0, 0, {'G', {}}, nullptr }; PackageInfo shipment = { 0.0, 3.0, {15, 15} }; | 1 | 1 | PASS |
| Test data for T4 (Invalid shipment: invalid destination row) | Truck truck = { 4, 1500.0, 20.0, 0, 0, {'B', {}}, nullptr }; PackageInfo shipment = { 1200.0, 5.0, {30, 10} }; | 1 | 1 | PASS |

**TEST NAME OR ID**: T5-T8

**Test Type**: Integration Testing

**Description**: Test the internal logic of the isTruckCapacitySufficient function, including loop iterations, conditional paths, and updates.

**Setup:**

**Test Function**:

LimitingFactor()

CompareRemaining()

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Truck with more remaining weight capacity than volume | Truck truck1 = { 1, 1500.0, 25.0, 0, 0, {'B', {}}, nullptr };  Truck truck2 = { 2, 1200.0, 30.0, 0, 0, {'Y', {}}, nullptr }; | -1 | -1 | PASS |
| Truck with more remaining volume capacity than weight | Truck truck1 = { 3, 2000.0, 20.0, 0, 0, {'G', {}}, nullptr };  Truck truck2 = { 4, 1500.0, 25.0, 0, 0, {'B', {}}, nullptr }; | 1 | 1 | PASS |
| Both trucks with equal remaining capacities | Truck truck1 = { 5, 1000.0, 30.0, 0, 0, {'Y', {}}, nullptr };  Truck truck2 = { 6, 1000.0, 30.0, 0, 0, {'G', {}}, nullptr }; | 1 | 1 | PASS |
| One truck with remaining weight capacity being the limiting factor. | Truck truck1 = { 7, 2000.0, 10.0, 0, 0, {'B', {}}, nullptr };  Truck truck2 = { 8, 1500.0, 30.0, 0, 0, {'G', {}}, nullptr }; | -1 | -1 | PASS |

**TEST NAME OR ID**: T9-T12

**Test Type**: Integration Testing

**Description**: Test the internal logic of the assignShipmentToTruck () after the shipment is validated.

**Setup:** VS code unit test template, functions and a main() to execute the test function.

**Test Function**:

validateShipment()

assignShipmentToTruck()

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Create a shipment that is valid | Map map = populateMap();  PackageInfo shipment = { 600.0, 20.0, {3, 3} }; Route route1 = { {{1, 1}, {2, 2}, {3, 3}}, 3, 'B' };  Truck truck1 = { 1, 2500.0f, 100.0f, 0, 0, route1, nullptr };  Route route2 = { {{4, 4}, {5, 5}}, 2, 'G' }; Truck truck2 = { 2, 1500.0f, 50.0f, 0, 0, route2, nullptr };  Truck fleet[] = { truck1, truck2 }; int totalTrucks = 2;  destination= {4,4} | 0 | 0 | PASS |
| Create a shipment that exceeds weight capacity | PackageInfo shipment = { 3000.0, 20.0, {6, 6} };  Route route1 = { {{1, 1}, {2, 2}}, 2, 'B' }; Truck truck1 = { 1, 2500.0f, 100.0f, 0, 0, route1, nullptr };  Route route2 = { {{3, 3}, {4, 4}}, 2, 'G' }; Truck truck2 = { 2, 2000.0f, 80.0f, 0, 0, route2, nullptr };  Truck fleet[] = { truck1, truck2 }; int totalTrucks = 2; | -1 | -1 | PASS |
| No Trucks Available | Map map; Truck fleet[1] = { Truck(1, 5000) }; int totalTrucks = 0; | -1 | -1 | PASS |
| No truck should be assigned to an invalid shipment. | Truck fleet[2] = { Truck(1, 5000), Truck(2, 6000) }; int totalTrucks = 2; | -1 | -1 | PASS |

**TEST NAME OR ID**: T13-T16

**Test Type**: Integration Testing

**Description**: Test if the capacity in the truck is sufficient for further shipments and initialize the truck afterward.

**Setup:** VS code integration test template, functions, and a main() to execute the test function.

**Test Function**:

isTruckCapacitySufficient()

initializeTruck()

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Truck has just enough capacity for the package | Route route1 = { {{1, 1}, {1, 2}, {1, 3}}, 3, 'B' };  PackageInfo package1 = { 600.0, 3.0, {1, 1} }; PackageInfo\* packageArray[1] = { &package1 };  Truck truck1 = { 1, 2500.0f, 100.0f, 0, 1, route1, packageArray }; | 0 | 0 | PASS |
| Truck does not have enough capacity for the second package | Route route2 = { {{2, 1}, {2, 2}, {2, 3}}, 3, 'G' };  Truck truck2 = { 2, 1000.0f, 50.0f, 1, 0, route2, nullptr }; PackageInfo package2a = { 600.0f, 30.0f }; PackageInfo package2b = { 500.0f, 25.0f }; | 0 | 0 | PASS |
| Truck just barely cannot handle the weight of a new package | Route route3 = { {{3, 1}, {3, 2}, {3, 3}}, 3, 'Y' };  Truck truck3 = { 3, 600.0f, 50.0f, 1, 0, route3, nullptr }; PackageInfo package3a = { 500.0f, 25.0f }; PackageInfo package3b = { 200.0f, 20.0f }; | 0 | 0 | PASS |
| Truck cannot handle volume but can handle weight | Route route4 = { {{4, 1}, {4, 2}, {4, 3}}, 3, 'B' };  Truck truck4 = { 4, 2000.0f, 30.0f, 1, 0, route4, nullptr }; PackageInfo package4a = { 800.0f, 20.0f }; PackageInfo package4b = { 500.0f, 15.0f }; | 0 | 0 | PASS |

**TEST NAME OR ID**: T17-T20

**Test Type**: Integration Testing

**Description**: Test the function’s behavior after initializing the truck and further updating the truck’s capacity

**Setup:** VS code integration test template, functions, and a main() to execute the test function.

**Test Function**:

initializeTruck ()

updateTruckCapacity ()

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Initialize truck and then update its capacity | Map map = populateMap(); PackageInfo package1 = { 600.0, 10.0, {1, 1} }; PackageInfo package2 = { 500.0, 15.0, {2, 2} }; PackageInfo\* packageArray[10] = { nullptr };  // Initialize a truck Truck truck1 = { 1, 2500.0f, 100.0f, 0, 0, { { {1, 1}, {2, 2} }, 2, 'B' }, packageArray }; | 1 | 1 | PASS |
| Truck initialization with no packages and update capacity with no packages | Map map = populateMap();  // Create a truck with no initial packages PackageInfo\* emptyPackages[10] = { nullptr }; Truck truck2 = { 2, 2500.0f, 100.0f, 0, 0, { { {1, 1}, {3, 3} }, 2, 'G' }, emptyPackages }; | 1 | 1 | PASS |
| Initialize truck with packages and update capacity | Map map = populateMap();  // Create packages PackageInfo package1 = { 600.0, 10.0, {1, 1} }; PackageInfo package2 = { 500.0, 20.0, {2, 2} }; PackageInfo\* packageArray[10] = { &package1, &package2 };  // Initialize a truck with a route Truck truck1 = { 3, 2500.0f, 100.0f, 0, 0, { { {1, 1}, {2, 2}, {3, 3} }, 3, 'B' }, packageArray }; | 1 | 1 | PASS |
| Initialize truck and try updating its capacity with packages that exceed its available capacity | Map map = populateMap();  // Create packages that exceed the truck's capacity PackageInfo largePackage1 = { 3000.0, 50.0, {1, 1} }; // Exceeds weight capacity PackageInfo largePackage2 = { 150.0, 70.0, {2, 2} }; // Exceeds volume capacity PackageInfo\* packageArray[10] = { nullptr };  // Initialize a truck with a route Truck truck2 = { 4, 1000.0f, 30.0f, 0, 0, { { {1, 1}, {2, 2}, {3, 3} }, 3, 'G' }, packageArray }; | 0 | 0 | PASS |
|  |  |  |  |  |