Test Cases

**Test Number** - 1

**Test Name** – Create order success flow

**Description** – Check the creation of a new order

**Traceability** – 1 Order Management: Creating success

3 Payment Processing: Payment success

4 Inventory Management: Inventory update success

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator ,RabbitMQ

**Pre Conditions** – Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ, Basket simulator connected to RabbitMQ, Payment simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.1) | new entity with OrderStatusID =1 created in DB in Ordering.orders table  Basket simulator receives a message from Ordering with Routing key : OrderStartedIntegrationEvent (Output Messages#2.2.1)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:  Update Order entity Set OrderStatusID =2 in DB  Catalog simulator receives a message from Ordering with Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#2.2.4) |
| 2 | Catalog simulator sends a message to RabbitMQ to queue:Ordering with Routing key: OrderStockConfirmedIntegrationEvent (Input Messages#2.1.2) | Update order entity set OrderStatusID = 3 in DB  Payment simulator receives a message from Ordering with Routing key:  OrderStatusChangedToStockConfirmedIntegrationEvent (Output Messages#2.2.3) |
| 3 | Payment simulator send a message to RabbitMQ to queue: Ordering with Routing key: OrderPaymentSucceededIntegrationEvent(Input Messages#2.1.6) | Update Order entity Set OrderStatusID =4 in DB  Catalog simulator receives a message from Ordering with Routing key:  OrderStatusChangedToPaidIntegrationEvent (Output Messages#2.2.5) |

**Test Number** - 2

**Test Name** – Create order out of stock failure flow

**Description** – Check the creation of a new order without enough items in stock

**Traceability** – 1 Order Management: Creating failure

4 Inventory Management :products are available to purchase

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator , RabbitMQ

**Pre Conditions** – Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ, Basket simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.1) | new entity with OrderStatusID =1 created in DB in Ordering.orders table  Basket simulator receives a message from Ordering with Routing key : OrderStartedIntegrationEvent (Output Messages#2.2.1)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:  Update Order entity Set OrderStatusID =2 in DB  Catalog simulator receives a message from Ordering with Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#2.2.4) |
| 2 | Catalog simulator Send a message to RabbitMQ to queue:Ordering with Routing key: OrderStockRejectedIntegrationEvent (Input Messages#2.1.3) | Update Order entity Set OrderStatusID = 6 in DB |

**Test Number** - 3

**Test Name** – Create order with payment failure

**Description** – Check the creation of a new order with payment failure

**Traceability** – 1 Order Management: Creating failure

3 Payment Processing: Payment failure

4 Inventory Management :products are available to purchase

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator , RabbitMQ

**Pre Conditions** – Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ, Basket simulator connected to RabbitMQ, Payment simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.1) | new entity with OrderStatusID =1 created in DB in Ordering.orders table  Basket simulator receives a message from Ordering with Routing key : OrderStartedIntegrationEvent (Output Messages#2.2.1)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:  Update Order entity Set OrderStatusID =2 in DB  Catalog simulator receives a message from Ordering with Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#2.2.4) |
| 2 | Catalog simulator sends a message to RabbitMQ to queue:Ordering with Routing key: OrderStockConfirmedIntegrationEvent (Input Messages#2.1.2) | Update order entity set OrderStatusID = 3 in DB  Payment simulator receives a message from Ordering with Routing key:  OrderStatusChangedToStockConfirmedIntegrationEvent (Output Messages#2.2.3) |
| 3 | payment simulator send a message to RabbitMQ to queue: Ordering with Routing key: OrderPaymentFailedIntegrationEvent (Input Messages#2.1.7) | Update Order entity Set OrderStatusID =6 in DB |

**Test Number** - 4

**Test Name –** Create an order with Quantity 0

**Description –** Check the creation of a new order when the Quantity =0

**Traceability –** 1 Order Management: Creating failure

4 Inventory Management :products are availableto purchase

**Environment –** Catalog simulator , Ordering service , Basket simulator , Payment simulator , RabbitMQ

**Pre Conditions –** Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ, Basket simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.1) | Basket simulator receives a message from Ordering with Routing key : OrderStartedIntegrationEvent (Output Messages#2.2.1)  New entity doesn't created in the DB |

**Test Number** - 5

**Test Name –** Create an order without items in the basket

**Description –** Check the creation of a new order when basket is empty

**Traceability –** 1 Order Management: Creating failure

4 Inventory Management :products are availableto purchase

**Environment –** Catalog simulator , Ordering service , Basket simulator , Payment simulator , RabbitMQ

**Pre Conditions –** Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ, Basket simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.1) with empty list of items | Basket simulator receives a message from Ordering with Routing key : OrderStartedIntegrationEvent (Output Messages#2.2.1)  New entity doesn't created in the DB |

**Test Number** - 6

**Test Name** – Update order success flow

**Description** – Check the update of an order from status 4 to 5

**Traceability** – 1 Order Management: Update success

3 Payment Processing: Payment success

4 Inventory Management :update success

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator ,RabbitMQ

**Pre Conditions** – Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ, Basket simulator connected to RabbitMQ, Payment simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.1) | new entity with OrderStatusID =1 created in DB in Ordering.orders table  Basket simulator receives a message from Ordering with Routing key : OrderStartedIntegrationEvent (Output Messages#1)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:  Update Order entity Set OrderStatusID =2 in DB  Catalog simulator receives a message from Ordering with Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#2.2.4) |
| 2 | Catalog simulator sends a message to RabbitMQ to queue:Ordering with Routing key: OrderStockConfirmedIntegrationEvent (Input Messages#2.1.2) | Update order entity set OrderStatusID = 3 in DB  Payment simulator receives a message from Ordering with Routing key:  OrderStatusChangedToStockConfirmedIntegrationEvent (Output Messages#2.2.3) |
| 3 | Payment simulator send a message to RabbitMQ to queue: Ordering with Routing key: OrderPaymentSucceededIntegrationEvent(Input Messages#2.1.6) | Update Order entity Set OrderStatusID =4 in DB  Catalog simulator receives a message from Ordering with Routing key:  OrderStatusChangedToPaidIntegrationEvent (Output Messages#2.2.5) |
| 4 | send a PUT(ship) message to Ordering (input message(x,y):   * **x-requestid**- X * **request body** - { "orderNumber": Y}) | Status code 200  Update Order entity Set OrderStatusID =5 in DB |

**Test Number** - 7

**Test Name** – Update order failure flow 1

**Description** – Check the update of an order from status 1 to 5

**Traceability** – 1 Order Management: Update failure

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator ,RabbitMQ

**Pre Conditions** – Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ, Ordering-Background turned off ,Basket simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.1) | new entity with OrderStatusID =1 created in DB in Ordering.orders table |
| 2 | send a PUT (ship) message to Ordering (input message(x,y):   * **x-requestid**- X * **request body** - { "orderNumber": Y}) | Status code 400  Order entity OrderStatusID stays on 1 in DB |

**Test Number** - 8

**Test Name** – Update order failure flow 2

**Description** – Check the update of an order from status 2 to 5

**Traceability** – 1 Order Management: Update failure

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator ,RabbitMQ

**Pre Conditions** – Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ, Basket simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.1) | new entity with OrderStatusID =1 created in DB in Ordering.orders table  Basket simulator receives a message from Ordering with Routing key : OrderStartedIntegrationEvent (Output Messages#1)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:  Update Order entity Set OrderStatusID =2 in DB  Catalog simulator receives a message from Ordering with Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#2.2.4) |
| 2 | send a PUT(ship) message to Ordering (input message(x,y):   * **x-requestid**- X * **request body** - { "orderNumber": Y   }) | Status code 400  Order entity OrderStatusID stays on 2 in DB |

**Test Number** - 9

**Test Name** – Update order failure flow 3

**Description** – Check the update of an order from status 3 to 5

**Traceability** – 1 Order Management: Update failure

4 Inventory Management : products are available to purchase

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator ,RabbitMQ

**Pre Conditions** – Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ, Basket simulator connected to RabbitMQ, , Payment simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.1) | new entity with OrderStatusID =1 created in DB in Ordering.orders table  Basket simulator receives a message from Ordering with Routing key : OrderStartedIntegrationEvent (Output Messages#2.2.1)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:  Update Order entity Set OrderStatusID =2 in DB  Catalog simulator receives a message from Ordering with Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#2.2.4) |
| 2 | Catalog simulator sends a message to RabbitMQ to queue:Ordering with Routing key: OrderStockConfirmedIntegrationEvent (Input Messages#2.1.2) | Update order entity set OrderStatusID = 3 in DB  Payment simulator receives a message from Ordering with Routing key:  OrderStatusChangedToStockConfirmedIntegrationEvent (Output Messages#2.2.3) |
| 3 | send a PUT(ship) message to Ordering (input message(x,y):   * **x-requestid**- X * **request body** - { "orderNumber": Y   }) | Status code 400  Order entity OrderStatusID stays on 3 in DB |

**Test Number** - 10

**Test Name** – Update order failure flow 4

**Description** – Check the update of an order from status 6 to 5

**Traceability** – 1 Order Management: Update failure

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator ,RabbitMQ

**Pre Conditions** – Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ, Basket simulator connected to RabbitMQ,

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.1) | new entity with OrderStatusID =1 created in DB in Ordering.orders table  Basket simulator receives a message from Ordering with Routing key : OrderStartedIntegrationEvent (Output Messages#1)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:  Update Order entity Set OrderStatusID =2 in DB  Catalog simulator receives a message from Ordering with Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#2.2.4) |
| 2 | Catalog simulator Send a message to RabbitMQ to queue:Ordering with Routing key: OrderStockRejectedIntegrationEvent (Input Messages#2.1.3) | Update Order entity Set OrderStatusID = 6 in DB |
| 3 | send a PUT (ship) message to Ordering (input message(x,y):   * **x-requestid**- X * **request body** - { "orderNumber": Y   }) | Status code 400  Order entity OrderStatusID stays on 6 in DB |

**Test Number** - 11

**Test Name** – Cancel order success 1

**Description** – Check the cancel of an order from status 1 to 6

**Traceability** – 1 Order Management: Cancel success

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator ,RabbitMQ

**Pre Conditions** – Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ, Ordering-background turned off ,Basket simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.1) | new entity with OrderStatusID =1 created in DB in Ordering.orders table |
| 2 | send a PUT(cancel) message to Ordering (input message(x,y):   * **x-requestid**- X * **request body** - { "orderNumber": Y   }) | Status code 200  Order entity OrderStatusID change to 6 in DB |

**Test Number** - 12

**Test Name** – Cancel order success 2

**Description** – Check the cancel of an order from status 2 to 6

**Traceability** – 1 Order Management: Cancel success

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator ,RabbitMQ

**Pre Conditions** – Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ,Basket simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.1) | new entity with OrderStatusID =1 created in DB in Ordering.orders table  Basket simulator receives a message from Ordering with Routing key : OrderStartedIntegrationEvent (Output Messages#1)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:  Update Order entity Set OrderStatusID =2 in DB  Catalog simulator receives a message from Ordering with Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#2.2.4) |
| 2 | send a PUT(cancel) message to Ordering (input message(x,y):   * **x-requestid**- X * **request body** - { "orderNumber": Y   }) | Status code 200  Order entity OrderStatusID change to 6 in DB |

**Test Number** - 13

**Test Name** – Cancel order success 3

**Description** – Check the cancel of an order from status 3 to 6

**Traceability** – 1 Order Management: Cancel success

4 Inventory Management: products are available to purchase

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator ,RabbitMQ

**Pre Conditions** – Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ, Basket simulator connected to RabbitMQ, Payment simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.1) | new entity with OrderStatusID =1 created in DB in Ordering.orders table  Basket simulator receives a message from Ordering with Routing key : OrderStartedIntegrationEvent (Output Messages#1)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:  Update Order entity Set OrderStatusID =2 in DB  Catalog simulator receives a message from Ordering with Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#2.2.4) |
| 2 | Catalog simulator sends a message to RabbitMQ to queue:Ordering with Routing key: OrderStockConfirmedIntegrationEvent (Input Messages#2.1.2) | Update order entity set OrderStatusID = 3 in DB  Payment simulator receives a message from Ordering with Routing key:  OrderStatusChangedToStockConfirmedIntegrationEvent (Output Messages#2.2.3) |
| 3 | send a PUT(cancel) message to Ordering (input message(x,y):   * **x-requestid**- X * **request body** - { "orderNumber": Y   }) | Status code 200  Order entity OrderStatusID change to 6 in DB |

**Test Number** - 14

**Test Name** – Cancel order failure 1

**Description** – Check the cancel of an order from status 4 to 6

**Traceability** – 1 Order Management: Cancel failure

3 Payment Processing: Payment success

4 Inventory Management: update success

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator ,RabbitMQ

**Pre Conditions** – Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ, Basket simulator connected to RabbitMQ, , Payment simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.1) | new entity with OrderStatusID =1 created in DB in Ordering.orders table  Basket simulator receives a message from Ordering with Routing key : OrderStartedIntegrationEvent (Output Messages#1)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:  Update Order entity Set OrderStatusID =2 in DB  Catalog simulator receives a message from Ordering with Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#2.2.4) |
| 2 | Catalog simulator sends a message to RabbitMQ to queue:Ordering with Routing key: OrderStockConfirmedIntegrationEvent (Input Messages#2.1.2) | Update order entity set OrderStatusID = 3 in DB  Payment simulator receives a message from Ordering with Routing key:  OrderStatusChangedToStockConfirmedIntegrationEvent (Output Messages#2.2.3) |
| 3 | Payment simulator send a message to RabbitMQ to queue: Ordering with Routing key: OrderPaymentSucceededIntegrationEvent(Input Messages#2.1.6) | Update Order entity Set OrderStatusID =4 in DB  Catalog simulator receives a message from Ordering with Routing key:  OrderStatusChangedToPaidIntegrationEvent (Output Messages#2.2.5) |
| 4 | send a PUT(cancel) message to Ordering (input message(x,y):   * **x-requestid**- X * **request body** - { "orderNumber": Y}) | Status code 400  Order entity OrderStatusID stays on 4 in DB |

**Test Number** - 15

**Test Name** – Cancel order failure 2

**Description** – Check the cancel of an order from status 5 to 6

**Traceability** – 1 Order Management: Cancel failure

3 Payment Processing: Payment success

4 Inventory Management: update success

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator ,RabbitMQ

**Pre Conditions** – Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ, Basket simulator connected to RabbitMQ, , Payment simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.1) | new entity with OrderStatusID =1 created in DB in Ordering.orders table  Basket simulator receives a message from Ordering with Routing key : OrderStartedIntegrationEvent (Output Messages#1)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:  Update Order entity Set OrderStatusID =2 in DB  Catalog simulator receives a message from Ordering with Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#2.2.4) |
| 2 | Catalog simulator sends a message to RabbitMQ to queue:Ordering with Routing key: OrderStockConfirmedIntegrationEvent (Input Messages#2.1.2) | Update order entity set OrderStatusID = 3 in DB  Payment simulator receives a message from Ordering with Routing key:  OrderStatusChangedToStockConfirmedIntegrationEvent (Output Messages#2.2.3) |
| 3 | Payment simulator send a message to RabbitMQ to queue: Ordering with Routing key: OrderPaymentSucceededIntegrationEvent(Input Messages#2.1.6) | Update Order entity Set OrderStatusID =4 in DB  Catalog simulator receives a message from Ordering with Routing key:  OrderStatusChangedToPaidIntegrationEvent (Output Messages#2.2.5) |
| 4 | send a PUT(ship) message to Ordering (input message(x,y):   * **x-requestid**- X   **request body** - { "orderNumber": Y}) | Status code 200  Update Order entity Set OrderStatusID =5 in DB |
| 5 | send a PUT(cancel) message to Ordering (input message(x,y):   * **x-requestid**- X * **request body** - { "orderNumber": Y   }) | Status code 400  Order entity OrderStatusID stays on 5 in DB |

**Test Number** - 16

**Test Name** – Track of an order success flow 1

**Description** – Check that the order with status 1 is trackable

**Traceability** – 1 Order Management: create

2 Order Tracking

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator ,RabbitMQ

**Pre Conditions** – Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ, Ordering-Background turned off ,Basket simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.1) | new entity with OrderStatusID =1 created in DB in Ordering.orders table |
| 2 | send a GET (orderid) message to Ordering (input message(x):   * **orderid**- X   ) | Status code 200  OrderStatusID =1 in DB |

**Test Number** - 17

**Test Name** – Track of an order success flow 2

**Description** – Check that the order with status 2 is trackable

**Traceability** –1 Order Management: create

2 Order Tracking

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator ,RabbitMQ

**Pre Conditions** – Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ,Basket simulator connected to RabbitMQ, Payment simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.1) | Basket simulator receives a message from Ordering with Routing key : OrderStartedIntegrationEvent (Output Messages#1)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:  Update Order entity Set OrderStatusID =2 in DB  Catalog simulator receives a message from Ordering with Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#2.2.4) |
| 2 | send a GET (orderid) message to Ordering (input message(x):   * **orderid**- X   ) | Status code 200  OrderStatusID =2 in DB |

**Test Number** - 18

**Test Name** – Track of an order success flow 3

**Description** – Check that the order with status 3 is trackable

**Traceability** – 1 Order Management: create

2 Order Tracking

4 Inventory Management: products are available to purchase

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator ,RabbitMQ

**Pre Conditions** – Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ,Basket simulator connected to RabbitMQ, Payment simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.1) | Basket simulator receives a message from Ordering with Routing key : OrderStartedIntegrationEvent (Output Messages#1)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:  Update Order entity Set OrderStatusID =2 in DB  Catalog simulator receives a message from Ordering with Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#2.2.4) |
| 2 | Catalog simulator sends a message to RabbitMQ to queue:Ordering with Routing key: OrderStockConfirmedIntegrationEvent (Input Messages#2.1.2) | Update order entity set OrderStatusID = 3 in DB  Payment simulator receives a message from Ordering with Routing key:  OrderStatusChangedToStockConfirmedIntegrationEvent (Output Messages#2.2.3) |
| 3 | send a GET (orderid) message to Ordering (input message(x):   * **orderid**- X   ) | Status code 200  OrderStatusID =3 in DB |

**Test Number** - 19

**Test Name** – Track of an order success flow 4

**Description** – Check that the order with status 4 is trackable

**Traceability** – 1 Order Management: create

2 Order Tracking

3 Payment Processing: Payment success

4 Inventory Management: update success

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator ,RabbitMQ

**Pre Conditions** – Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ,Basket simulator connected to RabbitMQ, Payment simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.1) | Basket simulator receives a message from Ordering with Routing key : OrderStartedIntegrationEvent (Output Messages#1)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:  Update Order entity Set OrderStatusID =2 in DB  Catalog simulator receives a message from Ordering with Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#2.2.4) |
| 2 | Catalog simulator sends a message to RabbitMQ to queue:Ordering with Routing key: OrderStockConfirmedIntegrationEvent (Input Messages#2.1.2) | Update order entity set OrderStatusID = 3 in DB  Payment simulator receives a message from Ordering with Routing key:  OrderStatusChangedToStockConfirmedIntegrationEvent (Output Messages#2.2.3) |
| 3 | Payment simulator send a message to RabbitMQ to queue: Ordering with Routing key: OrderPaymentSucceededIntegrationEvent(Input Messages#2.1.6) | Update Order entity Set OrderStatusID =4 in DB  Catalog simulator receives a message from Ordering with Routing key:  OrderStatusChangedToPaidIntegrationEvent (Output Messages#2.2.5) |
| 4 | send a GET (orderid) message to Ordering (input message(x):   * **orderid**- X   ) | Status code 200  OrderStatusID =4 in DB |

**Test Number** - 20

**Test Name** – Track of an order success flow 5

**Description** – Check that the order with status 5 is trackable

**Traceability** – 1 Order Management: create

2 Order Tracking

3 Payment Processing: Payment success

4 Inventory Management: update success

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator ,RabbitMQ

**Pre Conditions** – Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ,Basket simulator connected to RabbitMQ, Payment simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.1) | Basket simulator receives a message from Ordering with Routing key : OrderStartedIntegrationEvent (Output Messages#1)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:  Update Order entity Set OrderStatusID =2 in DB  Catalog simulator receives a message from Ordering with Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#2.2.4) |
| 2 | Catalog simulator sends a message to RabbitMQ to queue:Ordering with Routing key: OrderStockConfirmedIntegrationEvent (Input Messages#2.1.2) | Update order entity set OrderStatusID = 3 in DB  Payment simulator receives a message from Ordering with Routing key:  OrderStatusChangedToStockConfirmedIntegrationEvent (Output Messages#2.2.3) |
| 3 | Payment simulator send a message to RabbitMQ to queue: Ordering with Routing key: OrderPaymentSucceededIntegrationEvent(Input Messages#2.1.6) | Update Order entity Set OrderStatusID =4 in DB  Catalog simulator receives a message from Ordering with Routing key:  OrderStatusChangedToPaidIntegrationEvent (Output Messages#2.2.5) |
| 4 | send a PUT(ship) message to Ordering (input message(x,y):   * **x-requestid**- X   **request body** - { "orderNumber": Y}) | Status code 200  Update Order entity Set OrderStatusID =5 in DB |
| 5 | send a GET (orderid) message to Ordering (input message(x):   * **orderid**- X   ) | Status code 200  OrderStatusID =5 in DB |

**Test Number** - 21

**Test Name** – Track of an order success flow 6

**Description** – Check that the order with status 6 is trackable

**Traceability** – 1 Order Management: create

2 Order Tracking

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator ,RabbitMQ

**Pre Conditions** – Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ,Basket simulator connected to RabbitMQ, Payment simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.1) | new entity with OrderStatusID =1 created in DB in Ordering.orders table  Basket simulator receives a message from Ordering with Routing key : OrderStartedIntegrationEvent (Output Messages#1)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:  Update Order entity Set OrderStatusID =2 in DB  Catalog simulator receives a message from Ordering with Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#2.2.4) |
| 2 | send a PUT(cancel) message to Ordering (input message(x,y):   * **x-requestid**- X * **request body** - { "orderNumber": Y}) | Status code 200  Order entity OrderStatusID change to 6 in DB |
| 3 | send a GET (orderid) message to Ordering (input message(x):   * **orderid**- X   ) | Status code 200  OrderStatusID =6 in DB |

**Test Number** - 22

**Test Name** – Authentication success flow 1

**Description –** Check that the user cannot send a getrequest to an order of another user

**Traceability** – 5 Security

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator ,RabbitMQ

**Pre Conditions** – Bearer token connect to Alice user, Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ, Basket simulator connected to RabbitMQ, Payment simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.8) | new entity with OrderStatusID =1 created in DB in Ordering.orders table  Basket simulator receives a message from Ordering with Routing key : OrderStartedIntegrationEvent (Output Messages#2.2.1)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:  Update Order entity Set OrderStatusID =2 in DB  Catalog simulator receives a message from Ordering with Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#2.2.4) |
| 2 | send a GET (bob's\_order \_orderid)  message to Ordering (input message(x):   * **orderid**- X   ) | Status code 400 |

**Test Number** - 23

**Test Name** – Authentication success flow 2

**Description –** Check that the user send a getrequest for all orders gets only his orders

**Traceability** – 5 Security

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator ,RabbitMQ

**Pre Conditions** – Bearer token connect to Alice user, Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ, Basket simulator connected to RabbitMQ, Payment simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.8) | new entity with OrderStatusID =1 created in DB in Ordering.orders table  Basket simulator receives a message from Ordering with Routing key : OrderStartedIntegrationEvent (Output Messages#2.2.1)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:  Update Order entity Set OrderStatusID =2 in DB  Catalog simulator receives a message from Ordering with Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#2.2.4) |
| 2 | send a GET orders  message to Ordering message() | Status code 200  Only Bob's orders displayed |

**Test Number** - 24

**Test Name** – Authentication success flow 3

**Description –** Check that the user cannot send a putrequest(ship) to an order of another user

**Traceability** – 5 Security

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator ,RabbitMQ

**Pre Conditions** – Bearer token connect to Alice user, Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ, Basket simulator connected to RabbitMQ, Payment simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.8) | new entity with OrderStatusID =1 created in DB in Ordering.orders table  Basket simulator receives a message from Ordering with Routing key : OrderStartedIntegrationEvent (Output Messages#2.2.1)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:  Update Order entity Set OrderStatusID =2 in DB  Catalog simulator receives a message from Ordering with Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#2.2.4) |
| 2 | send a PUT(ship) message to Ordering (input message(x,bob's\_order\_orderid):   * **x-requestid**- X   **request body** - { "orderNumber": bob's\_order\_orderid }) | Status code 400 |

**Test Number** - 25

**Test Name** – Authentication success flow 4

**Description –** Check that the user cannot send a putrequest(cancel) to an order of another user

**Traceability** – 5 Security

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator ,RabbitMQ

**Pre Conditions** – Bearer token connect to Alice user, Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ, Basket simulator connected to RabbitMQ, Payment simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.8) | new entity with OrderStatusID =1 created in DB in Ordering.orders table  Basket simulator receives a message from Ordering with Routing key : OrderStartedIntegrationEvent (Output Messages#2.2.1)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:  Update Order entity Set OrderStatusID =2 in DB  Catalog simulator receives a message from Ordering with Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#2.2.4) |
| 2 | send a PUT(cancel) message to Ordering (input message(x,bob's\_order\_orderid):   * **x-requestid**- X   **request body** - { "orderNumber": bob's\_order\_orderid }) | Status code 400 |

**Test Number** - 26

**Test Name** – Create 100 orders less than 1 hour

**Description** – Check the creation time of a new 100 orders is not takes more than 1 hour

**Traceability** – 1 Order Management: Creating success,

3 Payment Processing: Payment success,

4 Inventory Management: Inventory update success,

6 Scalability

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator ,RabbitMQ

**Pre Conditions** – Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ, Basket simulator connected to RabbitMQ, Payment simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.1) | new entity with OrderStatusID =1 created in DB in Ordering.orders table  Basket simulator receives a message from Ordering with Routing key : OrderStartedIntegrationEvent (Output Messages#2.2.1)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:  Update Order entity Set OrderStatusID =2 in DB  Catalog simulator receives a message from Ordering with Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#2.2.4) |
| 2 | Repeat the first step 100 times | 1 hour doesn’t pass yet |
| 3 | Catalog simulator sends a message to RabbitMQ to queue:Ordering with Routing key: OrderStockConfirmedIntegrationEvent (Input Messages#2.1.2) | Update order entity set OrderStatusID = 3 in DB  Payment simulator receives a message from Ordering with Routing key:  OrderStatusChangedToStockConfirmedIntegrationEvent (Output Messages#2.2.3) |
| 4 | Payment simulator send a message to RabbitMQ to queue: Ordering with Routing key: OrderPaymentSucceededIntegrationEvent(Input Messages#2.1.6) | Update Order entity Set OrderStatusID =4 in DB  Catalog simulator receives a message from Ordering with Routing key:  OrderStatusChangedToPaidIntegrationEvent (Output Messages#2.2.5) |

**Test Number** - 27

**Test Name** –System recovery success flow 1

**Description** – Check the recovery of the system after crash from status 1

**Traceability** – 1 Order Management: Creating success

3 Payment Processing: Payment success

4 Inventory Management :update success

7 Reliability : success

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator , RabbitMQ

**Pre Conditions** – Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ, Ordering background turned off, Basket simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.1) | new entity with OrderStatusID =1 created in DB in Ordering.orders table |
| 2 | Turn on the ordering background service | Basket simulator receives a message from Ordering with Routing key : OrderStartedIntegrationEvent (Output Messages#1)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:  Update Order entity Set OrderStatusID =2 in DB  Catalog simulator receives a message from Ordering with Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#2.2.4) |
| 3 | Catalog simulator sends a message to RabbitMQ to queue:Ordering with Routing key: OrderStockConfirmedIntegrationEvent (Input Messages#2.1.2) | Update order entity set OrderStatusID = 3 in DB  Payment simulator receives a message from Ordering with Routing key:  OrderStatusChangedToStockConfirmedIntegrationEvent (Output Messages#2.2.3) |
| 4 | Payment simulator send a message to RabbitMQ to queue: Ordering with Routing key: OrderPaymentSucceededIntegrationEvent(Input Messages#2.1.6) | Update Order entity Set OrderStatusID =4 in DB  Catalog simulator receives a message from Ordering with Routing key:  OrderStatusChangedToPaidIntegrationEvent (Output Messages#2.2.5) |

**Test Number** - 28

**Test Name** –System recovery success flow 2

**Description** – Check the recovery of the system after crash from status 2

**Traceability** – 1 Order Management: Creating success

3 Payment Processing: Payment success

4 Inventory Management :update success

7 Reliability : success

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator , RabbitMQ

**Pre Conditions** – Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ, Basket simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.1) | new entity with OrderStatusID =1 created in DB in Ordering.orders table  Basket simulator receives a message from Ordering with Routing key : OrderStartedIntegrationEvent (Output Messages#1)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:  Update Order entity Set OrderStatusID =2 in DB  Catalog simulator receives a message from Ordering with Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#2.2.4) |
| 2 | Turn off the ordering service | OrderStatusID stays on 2 in the DB |
| 3 | Catalog simulator sends a message to RabbitMQ to queue:Ordering with Routing key: OrderStockConfirmedIntegrationEvent (Input Messages#2.1.2) | OrderStatusID stays on 2 in the DB |
| 4 | Turn on the ordering service | Update order entity set OrderStatusID = 3 in DB  Payment simulator receives a message from Ordering with Routing key:  OrderStatusChangedToStockConfirmedIntegrationEvent (Output Messages#2.2.3) |
| 5 | Payment simulator send a message to RabbitMQ to queue: Ordering with Routing key: OrderPaymentSucceededIntegrationEvent(Input Messages#2.1.6) | Update Order entity Set OrderStatusID =4 in DB  Catalog simulator receives a message from Ordering with Routing key:  OrderStatusChangedToPaidIntegrationEvent (Output Messages#2.2.5) |

**Test Number** - 29

**Test Name** –System recovery success flow 3

**Description** – Check the recovery of the system after crash from status 3

**Traceability** – 1 Order Management: Creating success

3 Payment Processing: Payment success

4 Inventory Management :update success

7 Reliability : success

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator , RabbitMQ

**Pre Conditions** – Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ, Basket simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.1) | new entity with OrderStatusID =1 created in DB in Ordering.orders table  Basket simulator receives a message from Ordering with Routing key : OrderStartedIntegrationEvent (Output Messages#1)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:  Update Order entity Set OrderStatusID =2 in DB  Catalog simulator receives a message from Ordering with Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#2.2.4) |
| 2 | Catalog simulator sends a message to RabbitMQ to queue:Ordering with Routing key: OrderStockConfirmedIntegrationEvent (Input Messages#2.1.2) | Update order entity set OrderStatusID = 3 in DB  Payment simulator receives a message from Ordering with Routing key:  OrderStatusChangedToStockConfirmedIntegrationEvent (Output Messages#2.2.3) |
| 3 | Turn off the ordering service | OrderStatusID stays on 3 in the DB |
| 4 | Payment simulator send a message to RabbitMQ to queue: Ordering with Routing key: OrderPaymentSucceededIntegrationEvent(Input Messages#2.1.6) | OrderStatusID stays on 3 in the DB |
| 5 | Turn on the ordering service | Update Order entity Set OrderStatusID =4 in DB  Catalog simulator receives a message from Ordering with Routing key:  OrderStatusChangedToPaidIntegrationEvent (Output Messages#2.2.5) |

**Test Number** - 30

**Test Name** – Authentication‏ success flow 1 bob user

**Description** – Check that the user(bob) cannot send a get request to an order of another user **(alice)**

**Traceability** – 5 Security

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator ,RabbitMQ

**Pre Conditions** – Bearer token connect to bob user,Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ, Basket simulator connected to RabbitMQ, Payment simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.1) | new entity with OrderStatusID =1 created in DB in Ordering.orders table  Basket simulator receives a message from Ordering with Routing key : OrderStartedIntegrationEvent (Output Messages#2.2.1)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:  Update Order entity Set OrderStatusID =2 in DB  Catalog simulator receives a message from Ordering with Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#2.2.4) |
| 2 | send a GET(orderID) message to Ordering with orderid of alice's order | Status code 400 |

**Test Number** - 31

**Test Name** – Authentication‏ success flow 2 bob user

**Description** – Check that the user(bob) cannot send a get orders request to an order of another user **(alice)**

**Traceability** – 5 Security

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator ,RabbitMQ

**Pre Conditions** – Bearer token connect to bob user,Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ, Basket simulator connected to RabbitMQ, Payment simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.1) | new entity with OrderStatusID =1 created in DB in Ordering.orders table  Basket simulator receives a message from Ordering with Routing key : OrderStartedIntegrationEvent (Output Messages#2.2.1)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:  Update Order entity Set OrderStatusID =2 in DB  Catalog simulator receives a message from Ordering with Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#2.2.4) |
| 2 | send a GET(orders) message | Status code 200  Only orders from bob are shown |

**Test Number** - 32

**Test Name** – Authentication‏ success flow 3 bob user

**Description** – Check that the user(bob) cannot send a put(ship) request to an order of another user **(alice)**

**Traceability** – 5 Security

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator ,RabbitMQ

**Pre Conditions** – Bearer token connect to bob user, Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ,Basket simulator connected to RabbitMQ, Payment simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.1) | new entity with OrderStatusID =1 created in DB in Ordering.orders table  Basket simulator receives a message from Ordering with Routing key : OrderStartedIntegrationEvent (Output Messages#2.2.1)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:  Update Order entity Set OrderStatusID =2 in DB  Catalog simulator receives a message from Ordering with Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#2.2.4) |
| 2 | send a PUT(ship) message to Ordering (input message(x,alice's\_order\_orderid):  • x-requestid- X  request body - { "orderNumber": alice's\_order\_orderid }) | Status code 400  Order entity OrderStatusID =4 in DB |

**Test Number** - 33

**Test Name** – Authentication‏ success flow 4 bob user

**Description** –– Check that the user(bob) cannot send a put(cancel) request to an order of another user **(alice)**

**Traceability** – 5 Security

**Environment** – Catalog simulator , Ordering service , Basket simulator , Payment simulator ,RabbitMQ

**Pre Conditions** – Bearer token connect to bob user, Catalog simulator connected to RabbitMQ , Ordering service connected to RabbitMQ,Basket simulator connected to RabbitMQ, simulator connected to RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **Steps** | **Expected Result** |
| 1 | Basket api simulator Sends a message to RabbitMQ to queue:Ordering with Routing key: UserCheckoutAcceptedIntegrationEvent (Input Messages#2.1.1) | new entity with OrderStatusID =1 created in DB in Ordering.orders table  Basket simulator receives a message from Ordering with Routing key : OrderStartedIntegrationEvent (Output Messages#2.2.1)  Ordering-Backgroudtasks find the record with OrderStatusID =1 and sends:  Update Order entity Set OrderStatusID =2 in DB  Catalog simulator receives a message from Ordering with Routing key: OrderStatusChangedToAwaitingValidationIntegrationEvent (Output Messages#2.2.4) |
| 2 | send a PUT(cancel) message to Ordering (input message(x,alice's\_order\_orderid):  • x-requestid- X  request body - { "orderNumber": alice's\_order\_orderid }) | Status code 400  Order entity OrderStatusID =2 in DB |