Business Process Automation Lab Demonstrator 2

(Guided Project)

Documentation for Order Management & Production Control Pools

in Winter 2023/24

TH Köln - Campus Gummersbach

Faculty of Computer Science and Engineering Science (F10)

Project by: Rahib Nazir Butt (11155035)

Berrak Kücük (11160144)

Project Supervisor: Prof. Dr. Matthias Zapp

Date: 1st of February, 2024

Contents

1. ORDER MANAGEMENT

Order Management pool handles the incoming customer orders. Process starts with receiving the order from the front-end application.

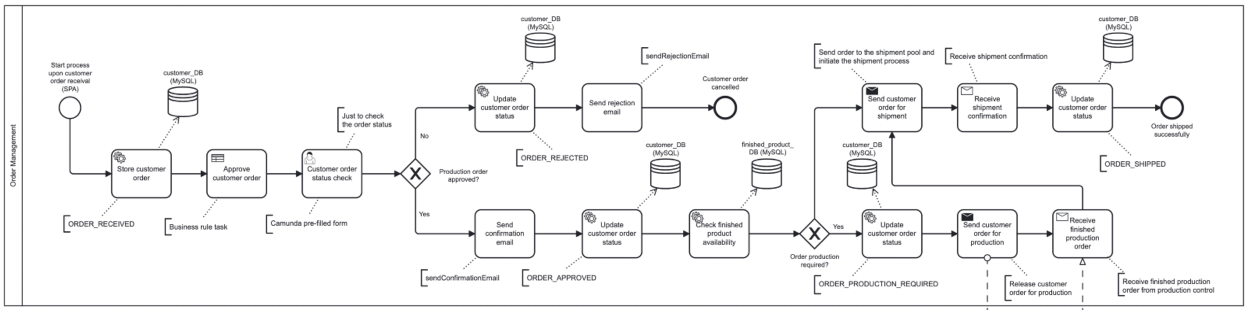
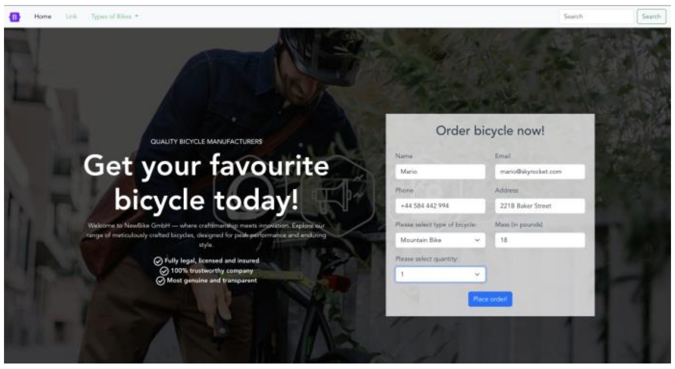

Order Management
Start process upon customer order receival (SPA)
customer_DB (MySQL)
Store customer order
ORDER_RECEIVED
Approve customer order
Business rule task
Customer order status check
Just to check the order status
Camunda pre-filled form
Production order approved?
No
Yes
customer_DB (MySQL)
Update customer order status
ORDER_REJECTED
Send rejection email
sendRejectionEmail
Customer order cancelled
Send confirmation email
sendConfirmationEmail
customer_DB (MySQL)
Update customer order status
ORDER_APPROVED
finished_product_ DB (MySQL)
Check finished product availability
Order production required?
Yes
Send order to the shipment pool and initiate the shipment process
Send customer order for shipment
Receive shipment confirmation
Receive shipment confirmation
customer_DB (MySQL)
Update customer order status
ORDER_SHIPPED
Order shipped successfully
customer_DB (MySQL)
Update customer order status
ORDER_PRODUCTION_REQUIRED
Send customer order for production
Release customer order for production
Receive finished production order
Receive finished production order from production control

Figure 1.1 Order Management


Home
Link
Types of Bikes
Search
Search
QUALITY BICYCLE MANUFACTURERS
Get your favourite bicycle today!
Welcome to NewBike GmbH — where craftsmanship meets innovation. Explore our range of meticulously crafted bicycles, designed for peak performance and enduring style.
Fully legal, licensed and insured
100% trustworthy company
Most genuine and transparent
Place order!

Order bicycle now!

Name: Mario
Email: mario@skyrocket.com
Phone: +44 584 442 994
Address: 221B Baker Street
Please select type of bicycle: Mountain Bike
Mass (in pounds): 18
Please select quantity: 1

Figure 1.2 Front-end application

As it can be seen in figure 1.1, there are many tasks and interactions with databases in this pool. List of the tasks and their purpose are listed below.

1.1 Store customer order (service task)

Customer orders are stored in the customer_DB database, inside of customer_order table. Each order has a unique id, product, and quantity, as well as customer information.

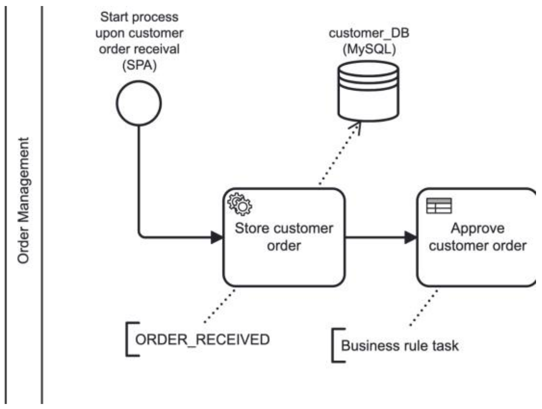
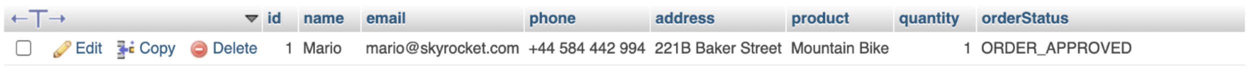


Figure 1.3 Store customer order service task



	id	name	email	phone	address	product	quantity	orderStatus
Edit Copy Delete	1	Mario	mario@skyrocket.com	+44 584 442 994	221B Baker Street	Mountain Bike	1	ORDER_APPROVED

Figure 1.4 customer_order table

1.2 Approve customer order (business rule task)

Depending on the quantity in the customer order, we categorize the order as either "single order" or "multiple order". If the quantity is greater than 5, than we cancel this order. In the first version, the decision was made with the DMN tables connected to each other, creating a DRD (Decision Requirements Diagrams). This can be seen in the figure 1.5. But we decided the complexity of this decision is not high, and we don't need to use DRD, a simple DMN table is enough (figure 1.6).

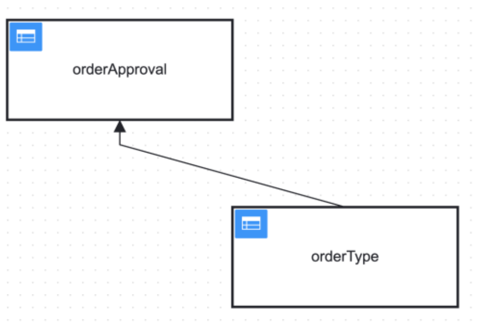
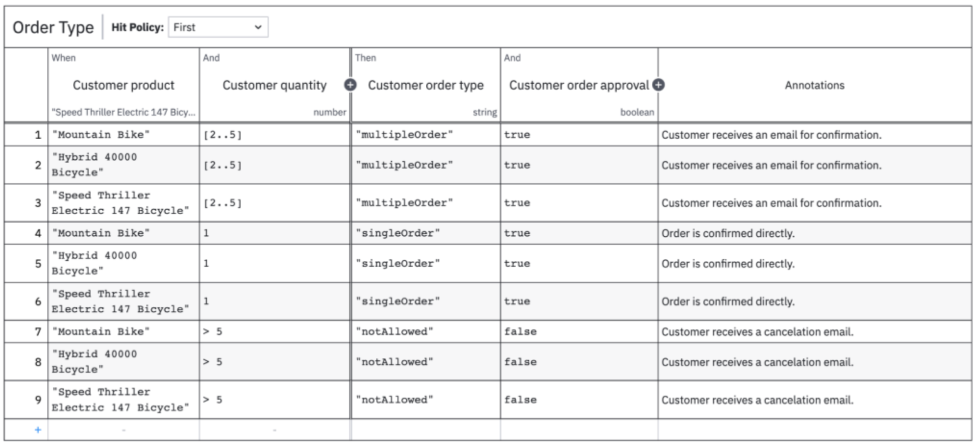


Figure 1.5 DRD (not in use)



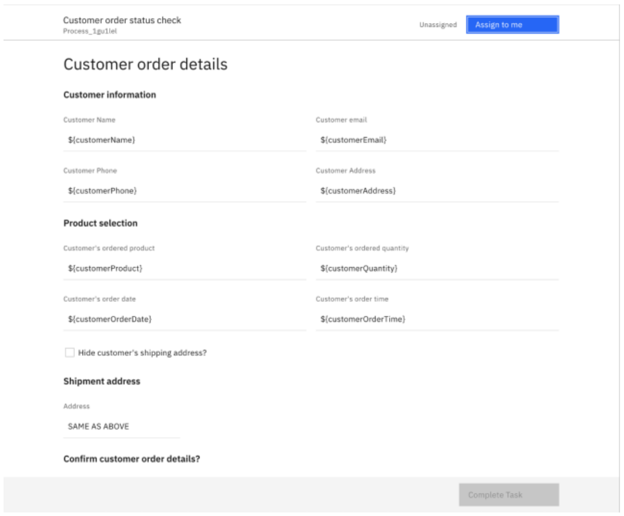
Order Type **Hit Policy:** First

	When	And	Then	And	
	Customer product	Customer quantity	Customer order type	Customer order approval	Annotations
	"Speed Thriller Electric 147 Bicy...	number	string	boolean	
1	"Mountain Bike"	[2..5]	"multipleOrder"	true	Customer receives an email for confirmation.
2	"Hybrid 40000 Bicycle"	[2..5]	"multipleOrder"	true	Customer receives an email for confirmation.
3	"Speed Thriller Electric 147 Bicycle"	[2..5]	"multipleOrder"	true	Customer receives an email for confirmation.
4	"Mountain Bike"	1	"singleOrder"	true	Order is confirmed directly.
5	"Hybrid 40000 Bicycle"	1	"singleOrder"	true	Order is confirmed directly.
6	"Speed Thriller Electric 147 Bicycle"	1	"singleOrder"	true	Order is confirmed directly.
7	"Mountain Bike"	> 5	"notAllowed"	false	Customer receives a cancelation email.
8	"Hybrid 40000 Bicycle"	> 5	"notAllowed"	false	Customer receives a cancelation email.
9	"Speed Thriller Electric 147 Bicycle"	> 5	"notAllowed"	false	Customer receives a cancelation email.
+	-	-			

Figure 1.6 DMN Table

1.3 Customer order status check (user task)

Here, the customer order can be viewed as a form, technically to slow down the process and see the order details together. This form takes the data from the customer order automatically, so there is no need to change or edit any field.



Customer order status check
Process_1gu1lel

Unassigned | Assign to me

Customer order details

Customer information

Customer Name: ${customerName}	Customer email: ${customerEmail}
Customer Phone: ${customerPhone}	Customer Address: ${customerAddress}

Product selection

Customer's ordered product: ${customerProduct}	Customer's ordered quantity: ${customerQuantity}
Customer's order date: ${customerOrderDate}	Customer's order time: ${customerOrderTime}

☐ Hide customer's shipping address?

Shipment address

Address: SAME AS ABOVE

Confirm customer order details?

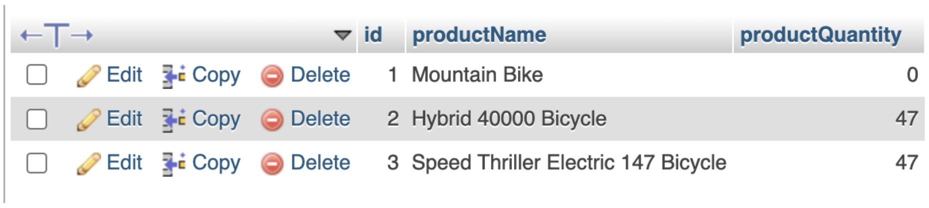
Complete Task

Figure 1.7 Customer order form

After the form, we have a gateway to check if the customer order is approved or not. If it is not approved, then we update the customer order status as ORDER_REJECTED and send a rejection email to the customer. If the order is approved, then we send a confirmation email to the customer that the order is received, and update customer order status as ORDER_APPROVED. After this step, we have to check if we have the product available.

1.4 Check finished product availability (service task)

We check from the finished_product_DB database if we have the ordered product available. Inside of the finished_product_stock, we have the product list with the available quantity.



				id	productName	productQuantity
☐	Edit	Copy	Delete	1	Mountain Bike	0
☐	Edit	Copy	Delete	2	Hybrid 40000 Bicycle	47
☐	Edit	Copy	Delete	3	Speed Thriller Electric 147 Bicycle	47

Figure 1.8 finished_product_stock table

Depending on the stock, we must go to the production or shipment step. Therefore we have another gateway. When we have enough stock, we send the customer order for shipment. This path connects with the Shipment pool by Send Task and Receive Task, which can be seen in figure 1.9. After shipment is done, customer order status is updated, and the process ends successfully since the order is shipped to the customer.

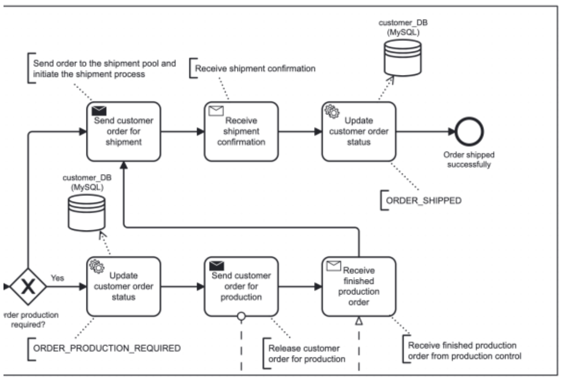

customer_DB (MySQL)
Send order to the shipment pool and initiate the shipment process
Receive shipment confirmation
Send customer order for shipment
Receive shipment confirmation
Update customer order status
Order shipped successfully
ORDER_SHIPPED
customer_DB (MySQL)
Yes
rder production required?
Update customer order status
Send customer order for production
Receive finished production order
ORDER_PRODUCTION_REQUIRED
Release customer order for production
Receive finished production order from production control

Figure 1.9

In the case that we don’t have enough stock, we update the order status as ORDER_PRODUCTION_REQUIRED and send the customer order for production. This continues in a different pool, “Production Control”, which will be explained in the second part. After the production is completed, we receive the finished production order and again continue with the shipment part.

2. PRODUCTION CONTROL

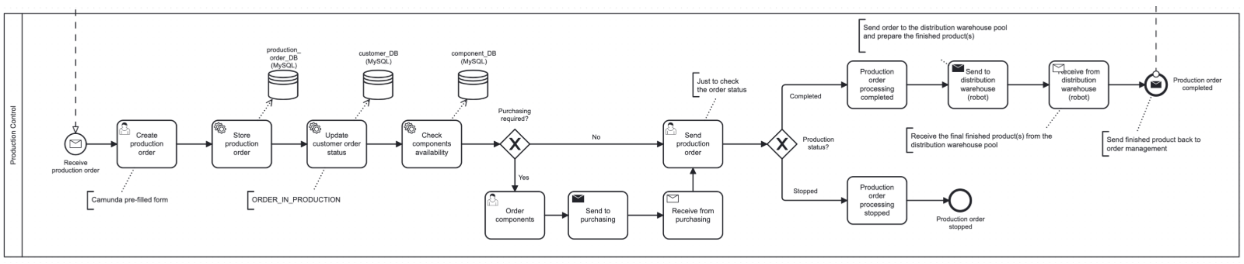
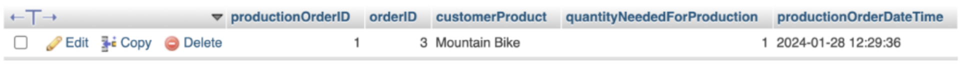

Production Control
Receive production order
Create production order
Camunda pre-filled form
Store production order
production_ order_DB (MySQL)
Update customer order status
customer_DB (MySQL)
ORDER_IN_PRODUCTION
Check components availability
component_DB (MySQL)
Purchasing required?
No
Yes
Order components
Send to purchasing
Receive from purchasing
Send production order
Just to check the order status
Production status?
Completed
Stopped
Production order processing completed
Send order to the distribution warehouse pool and prepare the finished product(s)
Send to distribution warehouse (robot)
Receive from distribution warehouse (robot)
Receive the final finished product(s) from the distribution warehouse pool
Production order completed
Send finished product back to order management
Production order processing stopped
Production order stopped

Figure 2.1

In the Production Control pool, process starts with receiving a production order from the Order Management pool. To see the production order before the process runs too fast, we have a User Task “Create production order”, which is a Camunda pre-filled form. Then we store the production order inside of production_order_DB database, as seen in the figure 2.2



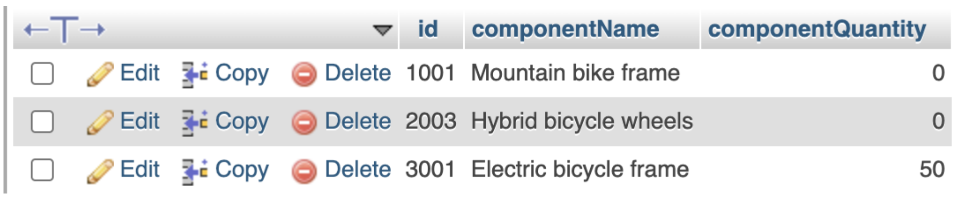
←T→	productionOrderID	orderID	customerProduct	quantityNeededForProduction	productionOrderDateTime
☐ Edit Copy Delete	1	3	Mountain Bike	1	2024-01-28 12:29:36

Figure 2.2 production_order table

After updating the customer order status as “ORDER_IN_PRODUCTION”, we need to check the component availability in order to continue with the production. We have available components listed inside component_DB database, and if there are missing components, we need to go to Purchasing. After purchasing is done, or if there is no need for purchasing, we start the production process. The implementation with the production robots is not done in this version, we have a gateway for checking the production status. Once it is completed, it sends the order to the distribution warehouse pool and prepares the finished product. Finally, the finished product is sent back to the Order Management Pool to be shipped to the customer.

2.1 Check Components Availability (service task)

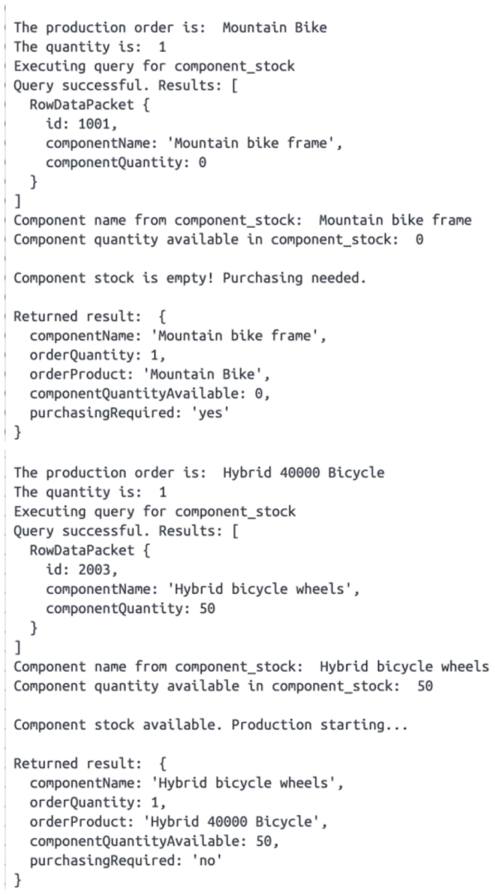
Inside of component_DB database, we have the table component_stock with the list of components and their quantities.



←T→	id	componentName	componentQuantity
☐ Edit Copy Delete	1001	Mountain bike frame	0
☐ Edit Copy Delete	2003	Hybrid bicycle wheels	0
☐ Edit Copy Delete	3001	Electric bicycle frame	50

Figure 2.3 component_stock table

This service task takes the most recent production order from production_order table (figure 2.2), and according to the name of the order, looks for the component. If the order is Mountain Bike, it will look for Mountain bike frame, or if it is Hybrid 40000 Bicycle, it will look for Hybrid bicycle wheels. If there is no stock of the component, it will go to Purchasing with the values like in figure 2.4 and when the stock is available it will go to Send Production Order by going through the gateway with "No".



```
The production order is:  Mountain Bike
The quantity is:  1
Executing query for component_stock
Query successful. Results: [
  RowDataPacket {
    id: 1001,
    componentName: 'Mountain bike frame',
    componentQuantity: 0
  }
]
Component name from component_stock:  Mountain bike frame
Component quantity available in component_stock:  0

Component stock is empty! Purchasing needed.

Returned result:  {
  componentName: 'Mountain bike frame',
  orderQuantity: 1,
  orderProduct: 'Mountain Bike',
  componentQuantityAvailable: 0,
  purchasingRequired: 'yes'
}

The production order is:  Hybrid 40000 Bicycle
The quantity is:  1
Executing query for component_stock
Query successful. Results: [
  RowDataPacket {
    id: 2003,
    componentName: 'Hybrid bicycle wheels',
    componentQuantity: 50
  }
]
Component name from component_stock:  Hybrid bicycle wheels
Component quantity available in component_stock:  50

Component stock available. Production starting...

Returned result:  {
  componentName: 'Hybrid bicycle wheels',
  orderQuantity: 1,
  orderProduct: 'Hybrid 40000 Bicycle',
  componentQuantityAvailable: 50,
  purchasingRequired: 'no'
}
```

Figure 2.4 Component Availability results

3. Test Cases

3.1 Single Order – Finished Product Availability

Selected bicycle type: Mountain Bike

Selected quantity: 1

Is there stock in the finished product database: Yes

Preconditions: Inside of the finished_product_stock table, there should be enough (in this case at least 1) productQuantity.

Expected results: Followed path should be

Order approved? - Yes

Order production required? - No

Goes to Shipment

Customer order status is updated

Process ends successfully with "Order shipped successfully"

Actual results: Same as the expected

Status: Passed

Notes: -

3.2 Single Order – Finished Product Not Available – Components Available

Selected bicycle type: Mountain Bike

Selected quantity: 1

Is there stock in the finished product database: No

Is there stock in the component database: Yes

Preconditions: Inside of the finished_product_stock table, there shouldn't be enough (in this case 0) productQuantity. Component of the selected bicycle type should have enough quantity (in this case Mountain bike frame, at least 1 quantity).

Expected results: Followed path should be

Order approved? - Yes

Order production required? - Yes

Customer order status updated (order production required)

Goes to Production Control

Production order is stored in the database

Purchasing required? - No

Production status – Completed

Production order completed

Goes back to Order Management

Goes to Shipment

Customer order status is updated

Process ends successfully with “Order shipped successfully”

Actual results: Same as expected

Status: Passed

Notes: -

3.3 Single Order – Finished Product Not Available – Components Not Available

Selected bicycle type: Mountain Bike

Selected quantity: 1

Preconditions: Inside of the finished_product_stock table, there shouldn’t be enough (in this case 0) productQuantity. Component of the selected bicycle type also should not have enough quantity (in this case Mountain bike frame, and 0 quantity).

Expected results: Followed path should be

Order approved? - Yes

Order production required? - Yes

Customer order status updated (order production required)

Goes to Production Control

Production order is stored in the database

Purchasing required? - Yes

Goes to Purchasing

Received back from Purchasing

Production status – Completed

Production order completed

Goes back to Order Management

Goes to Shipment

Customer order status is updated

Process ends successfully with “Order shipped successfully”

Actual results: Same as expected

Status: Passed

Notes: -

3.4 Single Order – Finished Product Not Available – Components Not Available

Selected bicycle type: Hybrid 40000 Bicycle

Selected quantity: 1

Is there stock in the finished product database: No

Is there stock in the component database: No

Preconditions: Inside of the finished_product_stock table, there shouldn’t be enough (in this case 0) productQuantity. Component of the selected bicycle type also should not have enough quantity (in this case Hybrid bicycle wheels, and 0 quantity).

Expected results: Followed path should be

> Order approved? - Yes
>
> Order production required? - Yes
>
> Customer order status updated (order production required)
>
> Goes to Production Control
>
> Production order is stored in the database
>
> Purchasing required? - Yes
>
> Goes to Purchasing
>
> Received back from Purchasing
>
> Production status – Completed
>
> Production order completed
>
> Goes back to Order Management
>
> Goes to Shipment
>
> Customer order status is updated
>
> Process ends successfully with “Order shipped successfully”

Actual results: Same as expected

Status: Passed

Notes: Logically, the amount for the purchase should be 2 instead of 1, since it is wheels. But this logic is not implemented and we send the amount to Purchasing as 1. It could be also thought of a pair of wheels, or it can be updated as “when it is wheels, multiply by 2”.

3.5 Multiple Order – Finished Product Available

Selected bicycle type: Mountain Bike

Selected quantity: 3

Is there stock in the finished product database: Yes

Preconditions: Inside of the finished_product_stock table, there should be enough (in this case at least 3) productQuantity.

Expected results: Followed path should be

Order approved? - Yes

Order production required? - No

Goes to Shipment

Customer order status is updated

Process ends successfully with "Order shipped successfully"

Actual results: Same as the expected

Status: Passed

Notes: -

3.6 Multiple Order – Quantity more than 5

Selected bicycle type: Mountain Bike

Selected quantity: More than 5

Is there stock in the finished product database: Yes

Preconditions: -

Expected results: Order should be rejected and followed path should be

Order approved? - No

Customer order status updated as "order rejected"

Process ends as "Customer order canceled"

Actual results: Same as the expected

Status: Passed

Notes: A solution for the Send Rejection Email task exists that triggers and generates an email to be sent to the customer regarding the current status of their order.

4. FURTHER DEVELOPMENTS

4.1 Checking multiple components

In the beginning we planned 3 components for each of the bicycles, however in this version, we are checking for one component for each bicycle. This can be improved by using a loop where all the different components are checked, and purchasing of the multiple components could be done.

4.2 Improving security including authentication

One of the improvements that could be done in terms of security is securing the access to the databases with a better authentication protocol. One of the protocols that we discussed was the Oauth which is an open standard used nowadays for the internet users of accessing any application or a server.

4.3 AI implementation

As you might be aware of how AI is progressing these days, we really thought of implementing AI in one of our processes. For example, implementing a chatbot somewhere in our factory model. Let’s say if we have a huge list of products, we could implement ChatGPT by OpenAI using the connector feature in order to generate good product descriptions after giving a specific prompt to the ChatGPT engine.

-----END-----