# Use Case: Manage orders

# Use case description for create orders

* Primary Actors: Admin
* Description: To enable the admin to create a new order, select a supplier, enter necessary details, and submit the order for further processing.
* Precondition:
  + The Admin must be logged in and have access to the orders module.
  + Products and Supplier Information are pre-loaded and available in the system.
* Postcondition: A new Order record is created and saved in the Orders table with the following details:
  + OrderID, OrderName, SupplierID, Shipping Date, Expected Delivery Date, Receipt Proof (if provided), Total Amount, Payment Status, and Comments.
  + The OrderStatus is set to "empty", indicating that the order can be modified with product details.
* Triggers:
  + Admin Accesses the order section and decides to manage orders

## Use case scenario on Adding a new order

Steps performed:

* + The admin starts the order creation process by selecting the "Create New Order" option in the admin interface.
  + The system either displays a modal with a new order form or redirects the admin to a new page with the order creation form.
  + The admin enters a **random order name** in the "Order Name" field.
  + The admin selects a supplier from the list of pre-existing suppliers, which links the supplier to the order via a **SupplierID**.
  + The admin selects the **shipping date** for the items via an input calendar.
  + The admin selects the expected delivery date for the order.
  + The admin uploads **receipt pictorials** showing proof of purchase for the order.
  + If no pictorials are available, the admin can enter a **raw text message** or another form of proof.
  + The admin enters the **total amount** for the order in the "Total Amount" field.
  + The admin selects a **payment status** (options: "fully paid", "partially paid", "pending", "completed").
  + The admin enters any **comments** related to the order in the comment field.
  + The admin clicks on the submit button to submit the order
  + The system checks if all **mandatory fields** are filled.
  + If any mandatory field is blank, the system returns a **304-status code** and prompts the admin to complete the form.
  + If all mandatory fields are filled, the system saves the order and returns a **success message** to confirm the order has been successfully created.
  + The order is marked with a **status of "empty"** (indicating it's ready for product details to be added).

#### **Alternative Flow(s)**:

1. **Mandatory Fields Missing**:
   * If any required field (e.g., **Order Name**, **Supplier**, **Total Amount**, etc.) is left blank, the system returns a **304-status** indicating that the form is incomplete, and the admin must correct the errors before submission.
2. **Admin Cancels Creation**:
   * If the admin decides not to create the order at any point, they can cancel the order creation process, and no data will be saved.

### Data structure

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Element name | Description | Length | Base  /Derived | Default value | Validation criteria | | |
|  |  |  |  |  | Upper limit | Lower limit | comments | |
| orderName | This is the name of the order | 10 | Base | Not null | Char =10 | Char >2 |  | |
| TotalAMount | Total amount of that order | base | Derived | Not null | Derived | derived |  | |
| paymentStatus | Payment status [‘paid’, ‘unpaid’, ‘partially paid’] | enum | Base | Not null | derived | Derived |  | |
| ShippingDate | Date order is shipped | Date | Base | Not null | Date | Date |  | |
| DeliveryDate | Date order is delivered | Date | Base | Not null | Date | Date |  | |
| Description | Random description of the order | Text | Base | Null | n/a | n/a |  | |
| Supplier details | Details of the supplier | Derived | Derived | Not null | n/a | n/a |  | |
| Receipt pictorials | Pictorials of payment receipts | Base | Base | null | Varchar(100) | Varchar(100) |  | |
| Receipt text | If no pictorials , raw text can be used | Varchar(255) | base | Null | Char=255 | Char=20 |  | |

### wireframe

### Dataflow

#### orderDetailstoDb dataflow

#### OrderDetailsFromDB datflow

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Element name | Description | Length | Base  /Derived | Default value | Validation criteria | | |
|  |  |  |  |  | Upper limit | Lower limit | comments | |
| orderName | This is the name of the order | 10 | Base | Not null | Char =10 | Char >2 |  | |
| TotalAMount | Total amount of that order | base | Derived | Not null | Derived | derived |  | |
| paymentStatus | Payment status [‘paid’, ‘unpaid’, ‘partially paid’] | enum | Base | Not null | derived | Derived |  | |
| ShippingDate | Date order is shipped | Date | Base | Not null | Date | Date |  | |
| DeliveryDate | Date order is delivered | Date | Base | Not null | Date | Date |  | |
| Description | Random description of the order | Text | Base | Null | n/a | n/a |  | |
| Supplier details | Details of the supplier | Derived | Derived | Not null | n/a | n/a |  | |
| Receipt imageurl | Pictorials of payment receipts | Base | Base | null | Varchar(100) | Varchar(100) |  | |
| Receipt text data | If no pictorials , raw text can be used | Varchar(255) | base | Null | Char=255 | Char=20 |  | |
| OrderId | ID for each order in the db | Varchar(25) | Derived | Derived | Char=30 | Char=20 |  | |
| CreatedAt | Date record was created | Date | Derived | Derived | Date | Date |  | |
| LastUpdatedAt | Date this record was updated | Date | Derived | Null | Date/null | Date/null |  | |
| Order status | Status[‘failed’,’pending’,’fulfilled’,’empty’,’not complete’] | Derived | Derived | Not null | n/a | n/a |  | |

### Schema design

## Use case scenario on editing an order

* Use case Name:
  + Edit order details

Primary Action: Admin

Description: To edit the details of an existing order, including modifying product items, quantities, or prices, while ensuring the **TotalAmount** in the **Orders Table** is consistent with the total of all associated product items.

Preconditions:

* The admin is logged into the system with appropriate permissions.
* The **Order** to be edited exists and is accessible by the admin.
* The order has a status that allows editing (e.g., **Pending** or Empty or Not complete).
* The order has at least one product item, if applicable.
* The system fetches the current details of the order, including product items, quantities, prices, and total order amount.
* Order fulfilled status is only set by the addition of product items to the stock db

Postconditions:

* The **Order Details** are updated based on admin input (including the **TotalAmount** field in the **Orders Table**).
* If there are product items in the order, the **TotalAmount** entered is compared against the total cost of those product items.
* If there are discrepancies between the **TotalAmount** and the sum of product items' costs, an error message is triggered.
* The order status remains unchanged (e.g., still **Pending** or **empty**) unless otherwise specified by the admin.
* The updated details are stored in the system, reflecting the correct amounts, product items, and statuses.

## Use case scenario on adding product details to order Details

* Use case Name:
  + Add product Details to order
* Primary Actor: Admin
* Description: To enable the admin to add product details to an existing order, including selecting products, specifying quantities, and entering prices.
* **Stakeholders and Interests**:
  + **Admin**: Needs to add products to the order in a clear, structured manner to ensure proper inventory tracking and order processing.
  + **System**: Ensures that only orders with the status **"empty"** can have products added, maintains data integrity in the **OrderDetails** table, and updates the **OrderStatus** accordingly.
* **Preconditions:**
* The **admin** is logged into the system.
* The **order** has been created with a status of **“empty”**.
* **Products** and **Units** are pre-loaded in the system and available for selection.
* The admin must be aware that product details can only be added to orders with a status of **“empty” or “pending”**.
* **Triggers**:
* The admin selects an existing **empty** order from the **Order Management** interface and chooses to add products.

Steps

* Admin Selects Order:
  + The admin navigates to the **Order Management** interface and selects an **empty** order to modify.
* System Displays Order Information:
  + The system displays a modal or page showing existing order details (if any) for that order.
* Admin Clicks on "Add Products to Order":
  + The admin clicks the "Add Products to Order" button.
* Admin Selects a Product:
  + The admin selects a **product** from a list of available pre-existing products, which links the product to the **OrderDetails** table via the **ProductID**.
* Admin Enters Quantity Ordered:
  + The admin specifies the **quantity** of the selected product being ordered.
* Admin Selects Unit of Quantity:
  + The admin selects the **unit** of quantity (e.g., piece, kg, liter) from a list of pre-existing units, linking the unit to the order via the **UnitID**.
* Admin Enters Price at Time of Order:
  + The admin enters the **price at time of order** for the selected product.
* Admin Submits Product Details:
  + The admin clicks **Submit** to add the product details to the order.
* System Updates Order:
  + The system adds a new record in the **OrderDetails** table with the following fields:

OrderDetailID, OrderID, ProductID, QuantityOrdered, UnitID, PriceAtTimeOfOrder, Subtotal (calculated as QuantityOrdered \* PriceAtTimeOfOrder).

* + The **OrderStatus** of the order is updated to **"pending"** to indicate that the order now contains products and is awaiting further processing.

#### **Alternative Flow(s)**:

1. **Product Not Available**:

* If the product selected is not available or out of stock, the system alerts the admin and prevents the product from being added to the order.

1. **Admin Modifies Order**:
   * If the admin changes the quantity or selects additional products, the **OrderDetails** table is updated accordingly, and the **TotalAmount** for the order is recalculated.

#### **Postconditions**:

* The **OrderDetails** table is updated with one or more product entries, including the **ProductID**, **QuantityOrdered**, **UnitID**, and **PriceAtTimeOfOrder**.
* The **OrderStatus** for the order is updated to **"pending"** to indicate that product details have been successfully added.

#### **Extensions and Exceptions**:

* **Order Status Not "Empty"**:
  + If the admin tries to add products to an order with a status other than **"empty" or “pending”** (e.g., "success", or "failed”), the system will prevent the modification and display an error message.
* **Mandatory Fields Missing**:
  + If any mandatory product details (e.g., **ProductID**, **Quantity**, **PriceAtTimeOfOrder**) are missing, the system will prompt the admin to complete the required fields.

### Data structure

### Wireframe

### Dataflow

#### Add products to order details table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Element name | Description | Length | Base  /Derived | Default value | Validation criteria | | |
|  |  |  |  |  | Upper limit | Lower limit | comments | |
| OrderId | This is the id of the product ordered | 10 | Base | Not null | Char =50 | Char >20 |  | |
| ProductSku | Unit sku for each product | derived | Derived | Not null | Derived | derived |  | |
| QuantityOrdered | In numeral form eg 10 | Base | Base | Not null | int<10000 | int>0 |  | |
| Units | Eg kgs | Derived | derived | Not null | derived | derived |  | |
| priceAtTimeOfOrder | Total price of that product item | >0 | Base | Not null | >0 | Float 3 |  | |
| dateAdded | Date order details is added | Date | Base | Not null | Date | Date |  | |
| dateLastUpdated | Date order is updated | Date | Base | null | Date | Date |  | |

#### fetch products from order details table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Element name | Description | Length | Base  /Derived | Default value | Validation criteria | | |
|  |  |  |  |  | Upper limit | Lower limit | comments | |
| OrderId | This is the id of the product ordered | 10 | Base | Not null | Char =50 | Char >20 |  | |
| orderDetailID | Id of the orderDetail | 10 | Derived | Not null | Char=50 | Char >20 |  | |
| ProductSku | Unit sku for each product | derived | Derived | Not null | Derived | derived |  | |
| productName |  |  |  |  |  |  |  | |
| QuantityOrdered | In numeral form eg 10 | Base | Base | Not null | int<10000 | int>0 |  | |
| Units | Eg kgs | Derived | derived | Not null | derived | derived |  | |
| priceAtTimeOfOrder | Total price of that product item | >0 | Base | Not null | >0 | Float 3 |  | |
| dateAdded | Date order details is added | Date | Base | Not null | Date | Date |  | |
| dateLastUpdated | Date order is updated | Date | Base | Not null | Date | Date |  | |

## Data Element Description Structure

|  |  |  |  |
| --- | --- | --- | --- |
| Data element | description | | |
| orderStatus[failed, fulfilled,empty, pending, not complete] |  | description | condition |
| failed | Order for whatever reasons did not complete | No items on this order have added to stock or order is empty status |
| fulfilled | All Order items have been successfully added to stock | Status turns to fulfilled once all product items in that order have been added to stock |
| Empty | Order has been created but no product items have been added to that order |  |
| Pending | Product items have been added but order has not arrived | One or more product items has been added to that stock and the delivery date is not < than today’s date |
| Not complete | Some product items have been added to order but not all |  |