MATERIAL ORDER

Technical Documentation

Microsoft Power Platform

Version	1.0.0.9
Platform	Power Platform
Components	Canvas App, Dataverse, Power Automate
Author	Patryk Lyszkowski

Table of Contents

1. Application Overview

Material Order is a material order management system built on Microsoft Power Platform.
The application automates the process from submitting a request, through project manager approval, to fulfillment by the supplier.

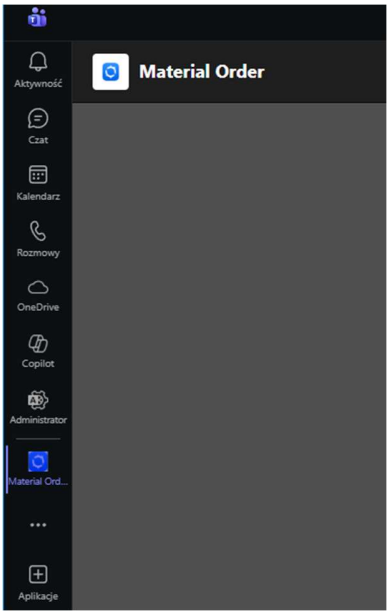
1.1. Core Functionality

The system supports:

- Multi-item material orders with quantity tracking
- Conditional approval processes based on project configuration
- Real-time notifications in Teams via Adaptive Cards
- Role-based access (Supervisor / Manager / Supplier)
- Full order lifecycle management

1.2. Architecture

Main access point: application embedded in Microsoft Teams.
Users interact with the Canvas App directly in Teams without installing Power Apps separately. The application uses Microsoft Teams for authentication and for delivering notifications.



Screenshot 1: Material Order in the left Teams panel – application available directly in the interface

1.3. Technical Stack

Warstwa	Technologia
Presentation	Canvas App in Teams

Business Logic	Power Automate cloud flows (automated triggers)
Data	Dataverse – relational database
Integration	Teams Connector (Adaptive Cards), Dataverse Connector

2. Data Model

The system uses four custom Dataverse tables.
All tables automatically track change history: creation and modification dates (CreatedOn, ModifiedOn), authors (CreatedBy, ModifiedBy), and owner (Owner).

2.1. Table: CompanyUser

Stores system users with role-based access control.

Column	Type	Required	Notes
name	Text (100)	Yes	Primary Name field
fullname	Text (200)	Yes	Display name
role	Choice	Yes	Employee / Manager / Supplier
isactive	Choice	No	Active / Inactive

2.2. Table: Project

Construction projects. Project settings determine whether orders require manager approval.

Column	Type	Required	Notes
name	Text (100)	Yes	Project identifier
address	Text (500)	Yes	Material delivery address
insurancecompany	Text (200)	No	Project name
requiresapproval	Two Options	Yes	Whether approval is required

2.3. Table: Order

Main material orders table. Tracks all fulfillment stages.

Column	Type	Required	Notes
name	Autonumber	Auto	Order number (ORD-{SEQNUM})
project	Lookup	Yes	→ project
status	Choice	Yes	7 states (see §2.6)
priority	Choice	Yes	Low/Medium/High/Urgent
requestedby	Lookup	Yes	→ companyuser (Employee)
assignedto	Lookup	Yes	→ companyuser (Supplier)
approvedby	Lookup	No	→ companyuser (Manager)
deliveryaddress	Text (500)	No	Inherited from project

2.4. Table: OrderItem

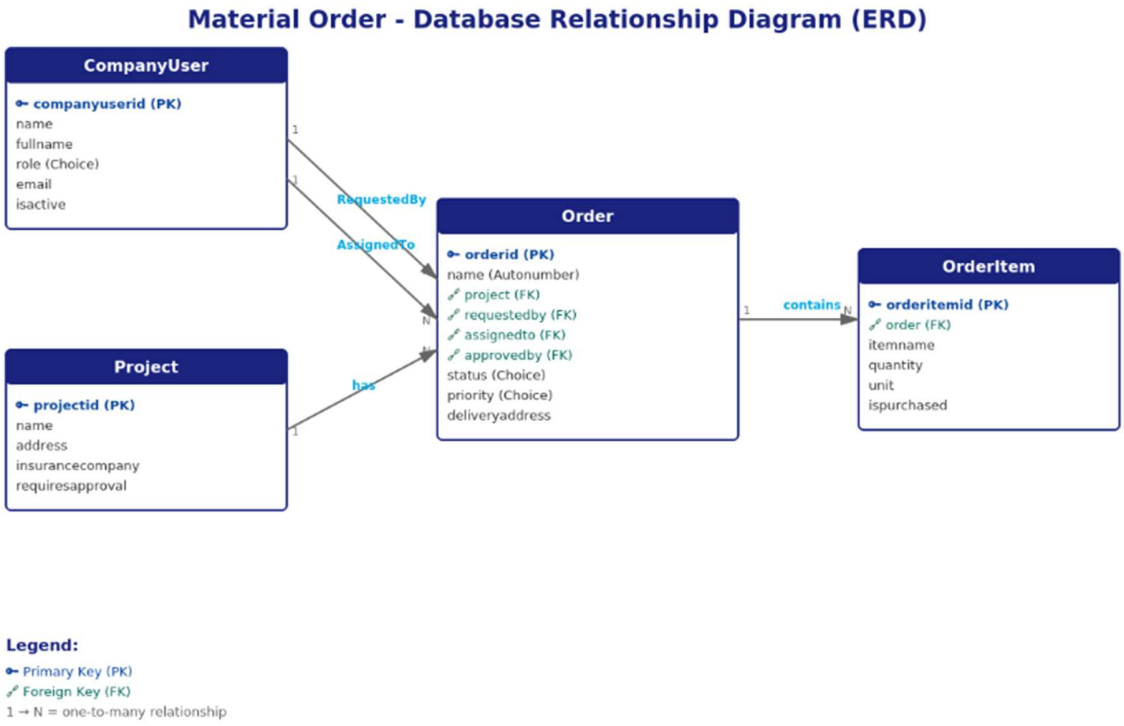
Order line items. Relationship: 1:N with Order.

Column	Type	Required	Notes
order	Lookup	Yes	→ order
itemname	Text (200)	Yes	Material description
quantity	Whole Number	Yes	Quantity
unit	Text (50)	Yes	kg, m, pcs., etc.
ispurchased	Two Options	No	Whether supplier purchased the material

2.5. Table Relationships

Key relationships:

From	To	Type	On delete
Project	Order	1:N	Restrict
Order	OrderItem	1:N	Cascade delete
CompanyUser	Order (RequestedBy)	1:N	Restrict
CompanyUser	Order (AssignedTo)	1:N	Restrict



Screenshot 2: Table relationship diagram in the database

2.6. Order Status Lifecycle

Orders move through the following stages:

Status	Opis
Draft	Initial state. Employee can edit.
Submitted	Sent for approval. Triggers workflow if project requires approval.
Approved	Manager approved. Ready for supplier assignment.

Rejected	Manager rejected.
Changes Requested	Manager requested modifications. Returns to Draft.
In Progress	Supplier is actively fulfilling the order.
Completed	All items purchased and delivered.

3. Power Automate Flows

The system uses two cloud flows with Dataverse triggers and Teams actions.

3.1. Flow: NewOrderAdded

Automatic flow triggered when an Order is created.

3.1.1. Trigger Configuration

Trigger: New record created in Orders table.
The flow runs immediately when an employee submits an order.

3.1.2. Workflow Logic

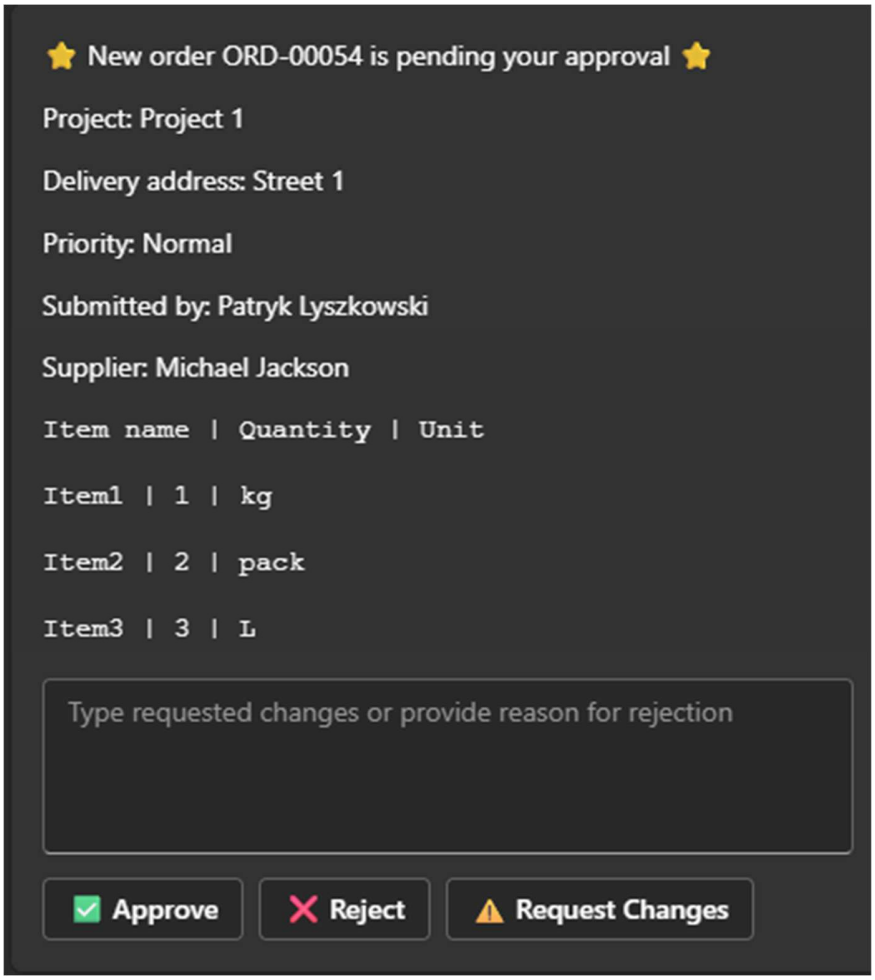
1. Retrieves project data and checks if approval is required (requiresapproval)
2. If yes – collects employee, supplier, and manager info
3. Loads all OrderItems
4. Formats material list into a readable table
5. Sends Adaptive Card to Teams with 3 actions: Approve / Reject / Request Changes

3.1.3. Adaptive Card Structure

The card contains dynamic content and three options to choose:

```
{
  "type": "AdaptiveCard",
  "body": [
    {
      "type": "TextBlock",
      "text": " OrderNumber ${OrderNumber}"
    },
    {
      "type": "TextBlock",
      "text": "Project: ${ProjectName}"
    },
    {
      "type": "TextBlock",
      "text": " Priority: ${Priority}"
    },
    {
      "type": "TextBlock",
      "text": "${ProductTable}"
    },
    {
      "type": "Input.Text",
```

```
      "id": "notes",
      "isMultiline": true
    }
  ],
  "actions": [
    {
      "type": "Action.Submit",
      "title": "Approve",
      "data": {"action": "approve"}
    },
    {
      "type": "Action.Submit",
      "title": "Reject",
      "data": {"action": "reject"}
    },
    {
      "type": "Action.Submit",
      "title": "Request Changes",
      "data": {"action": "change"}
    }
  ]
}
```



Screenshot 3: Adaptive Card in Teams – manager can approve, reject, or request changes

3.1.4. Action Handling

What happens after button click:

- Approve → Status = Approved, approval user and timestamp saved
- Reject → Status = Rejected, rejection reason saved
- Request Changes → Status returns to Draft with notes

3.2. Flow: OrderResubmitted

Handles resubmission after requested changes.

3.2.1. Trigger Configuration

1. Trigger: Order status changes to “Submitted”.
2. Flow runs when employee resubmits the corrected order.
3. Manager receives a new Teams notification.

3.2.2. Workflow Logic

Identical to NewOrderAdded but triggered only on resubmission, preventing duplicate notifications.

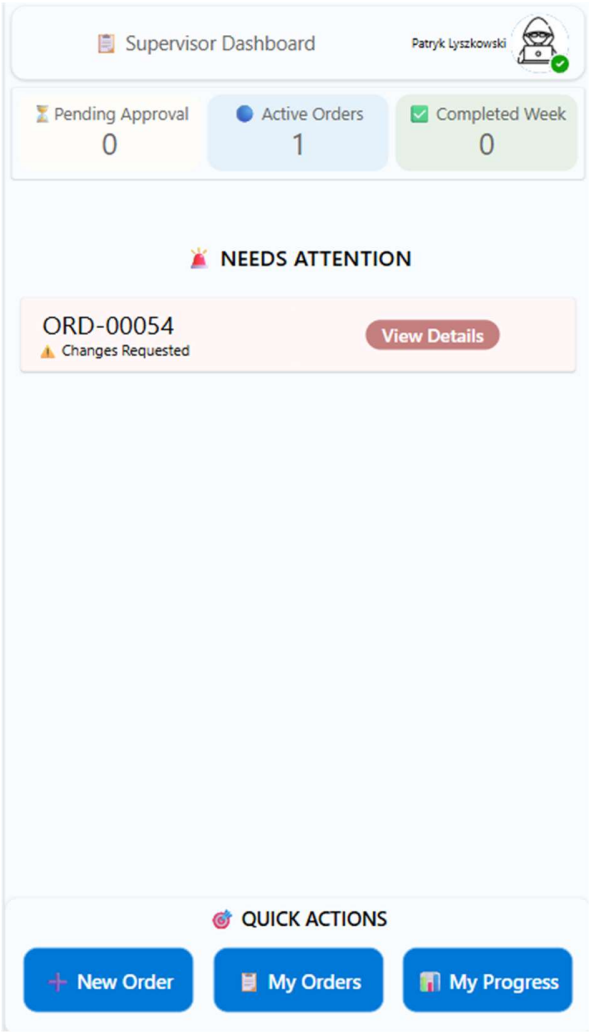
4. Material Order Application (Canvas App)

Canvas App embedded in Microsoft Teams.
Interface adapts to user role (Supervisor / Manager / Supplier).

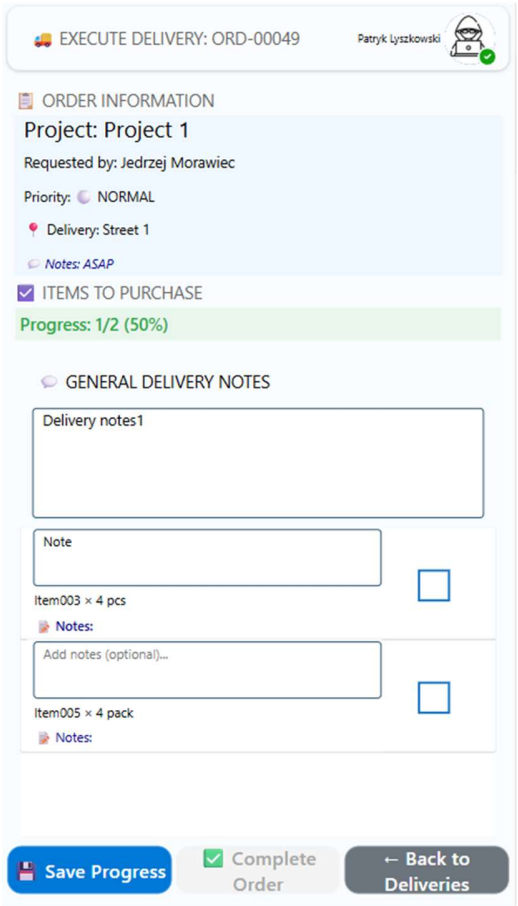
4.1. Application Architecture

4.1.1. Screens

- **HomeScreen:** Order statistics and navigation
- **NewOrderScreen:** Order creation form
- **OrderDetailScreen:** Order preview with history
- **SupplierExecuteScreen:** Fulfillment execution (Supplier only)
- **ProjectManagementScreen:** Project management (Manager only)



Screenshot 4: Home Screen – Supervisor dashboard



Screenshot 7: Supplier Screen – order fulfillment

The application also includes auxiliary screens, e.g., for user management and settings.

4.1.2. Global Variables

Example variables set at application startup:

```
Set(varCurrentUserEmail, User().Email);
Set(varCurrentUser,
    LookUp(
        CompanyUsers,
        Email = varCurrentUserEmail && IsActive = true
    )
);
// Permission check
If(
    IsBlank(varCurrentUser),
    // No access
    Set(varAccessDenied, true);
    Notify(
        "No access - Contact the Administrator",,
        NotificationType.Error
    ),
```

```
  // User role check
   Switch(
       varCurrentUser.Role,
       'Role (CompanyUsers)'.Supervisor,
           Set(varUserRole, "Supervisor"),
       'Role (CompanyUsers)'.Manager,
           Set(varUserRole, "Manager"),
       'Role (CompanyUsers)'.Supplier,
           Set(varUserRole, "Supplier"),
       Set(varUserRole, "Unknown")
   )
);
```

The application sets more than 30 global variables covering:

- Checking whether the user is active and has access to the system
- Blocking access with error messages for unauthorized users
- Converting system roles to readable text labels (Supervisor, Manager, Supplier)
- Full light/dark theme system with color palettes
- UI appearance variables (border colors, shadows, status colors)

4.2. Advanced PowerFX Patterns

For developers – examples of advanced techniques used in the application.

4.2.1. Role-based statistics

Home Screen OnVisible:

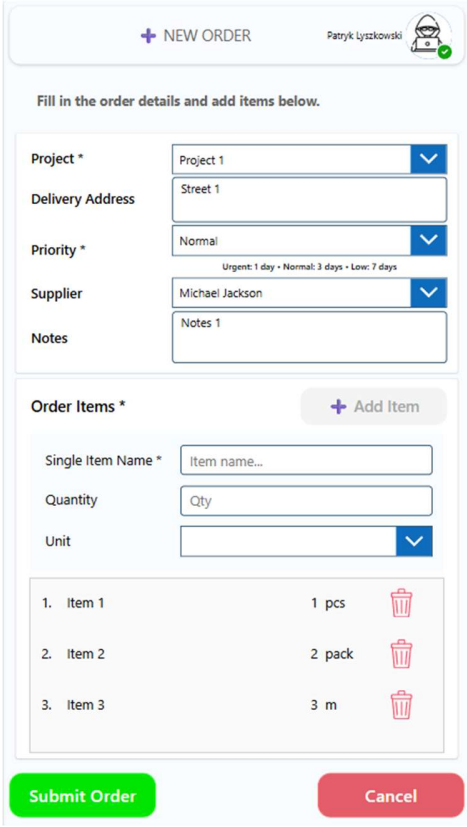
```
// Retrieve user with validation
Set(varCurrentUser,
    LookUp(
        CompanyUsers,
        Email = varCurrentUserEmail && IsActive = true
    )
);

// Convert Role enum to text
Switch(
    varCurrentUser.Role,
    'Role (CompanyUsers)'.Supervisor,
        Set(varUserRole, "Supervisor"),
    'Role (CompanyUsers)'.Manager,
        Set(varUserRole, "Manager"),
    'Role (CompanyUsers)'.Admin,
        Set(varUserRole, "Admin"),
    'Role (CompanyUsers)'.Supplier,
        Set(varUserRole, "Supplier"),
    Set(varUserRole, "Unknown")
);

// Statistics for Supervisor
If(
    varUserRole = "Supervisor",
    Set(varMyPendingApproval,
        CountRows(
```

```
            Filter(
                Orders,
                RequestedBy.Email = varCurrentUser.Email &&
                'Status (status)' = 'Status (Orders)'.'Pending Approval'
            )
        )
    );

    Set(varNeedsAttention,
        Filter(
            Orders,
            RequestedBy.Email = varCurrentUser.Email &&
            (
                'Status (status)' = 'Status (Orders)'.'Changes Requested' ||
                'Status (status)' = 'Status (Orders)'.Rejected ||
                (
                    'Status (status)' = 'Status (Orders)'.'Partially Completed' &&
                    StartedOn < varTwoDaysAgo
                )
            )
        )
    );
);
```

Solutions used:

- Converting roles from system format to readable text using Switch
- Automatic counting of orders with a specific status (CountRows + Filter)
- Complex filtering conditions – multiple checks in one expression
- Date checks – the system detects orders that are getting delayed

4.2.2. Creating an order with a materials list



Screenshot 6: New Order Form – adding materials

SupervisorNewOrder – Submit button:

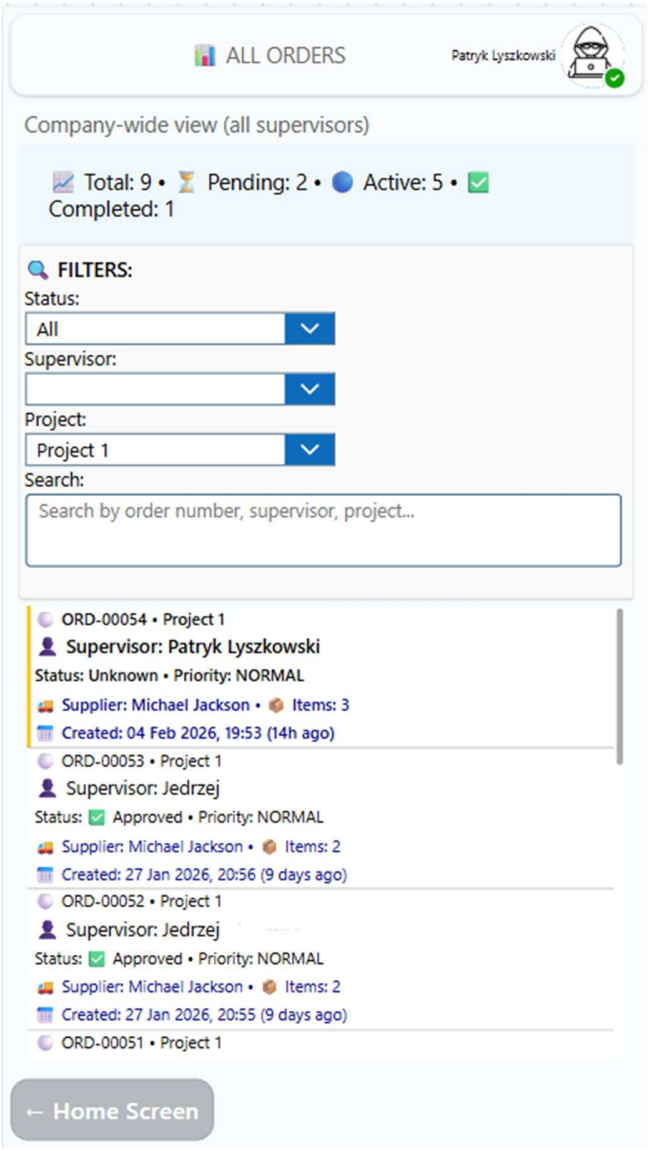
```
// STEP 1: Save the main order
Set(varNewOrder,
    Patch(
        Orders,
        Defaults(Orders),
        {
            Project: LookUp(Projects, Name = varSelectedProject.Name),
            RequestedBy: LookUp(CompanyUsers, Email = varCurrentUser.Email),
            AssignedTo: dd_NewOrderSupplier.Selected,
            Priority:
                Switch(
                    varSelectedPriority,
                    "Urgent", 'Priority (Orders)'.Urgent,
                    "Normal", 'Priority (Orders)'.Normal,
                    "Low", 'Priority (Orders)'.Low
                ),
            'Status (status)':
                If(
                    varSelectedProject.RequiresApproval,
                    'Status (Orders)'.'Pending Approval',
                    'Status (Orders)'.Approved
                ),
```

```
            SubmittedOn: Now()
        }
    )
);


// STEP 2: Save the materials list
ForAll(
    colOrderItems,
    Patch(
        OrderItems,
        Defaults(OrderItems),
        {
            Order: varNewOrder,
            ItemName: ItemName,
            Quantity: Quantity,
            Unit:
                Switch(
                    Unit,
                    "pcs", 'Unit (OrderItems)'.pcs,
                    "kg", 'Unit (OrderItems)'.kg,
                    "m", 'Unit (OrderItems)'.m,
                    "L", 'Unit (OrderItems)'.L
                ),
            IsPurchased: false
        }
    )
);

// STEP 3: Notification
Notify(
    "✅ Order " & varNewOrder.OrderNumber & " created! " &
    If(
        varSelectedProject.RequiresApproval,
        "Waiting for manager approval.",
        "Sent directly to supplier."
    ),
    NotificationType.Success,
    4000
);
```

Key elements:

- varNewOrder stores the created order record
- Data is retrieved safely using LookUp
- Automatic conversion from text values to system values
- Status depends on project configuration (approval required?)
- ForAll creates all line items in bulk
- Notification text changes dynamically

4.2.3. Order filtering with performance optimization



Screenshot 7: Orders list with filtering

ManagerAllOrdersScreen – Orders list with smart filtering:

```
SortByColumns(
    Filter(
        Filter(
            Orders,
            // First filter
            Switch(
```

```
                varFilterStatus,
                "All", true,
                "Pending Approval",
                    'Status (status)' = 'Status (Orders)'.'Pending Approval',
                "Approved",
                    'Status (status)' = 'Status (Orders)'.Approved,
                "In Progress",
                    'Status (status)' = 'Status (Orders)'.'In Progress',
                "Completed",
                    'Status (status)' = 'Status (Orders)'.Completed,
                true
            )
        ),
        // Second filter
        (varFilterSupervisor = "All" || RequestedBy.FullName = varFilterSupervisor) &&
        (varFilterProject = "All" || Project.Name = varFilterProject) &&
        (IsBlank(varOrderSearch) ||
         OrderNumber in varOrderSearch ||
         RequestedBy.FullName in varOrderSearch)
    ),
    "CreatedOn",
    Descending
)
```

How it works:

- Two filters applied sequentially
- First filter runs on the data source, second one in the app
- The first filter reduces the amount of data fetched from the server (faster)
- Switch for statuses instead of If – better performance
- Searching and filtering by user/project runs locally
- Safeguard against errors when the search field is empty
- Sorting by creation date (newest on top)

4.2.4. Adding and removing order items

SupervisorNewOrder – Managing order items:

```
// Initialize empty collection
Clear(colOrderItems);

// Add item (Add Item button)
Collect(
    colOrderItems,
    {
        ItemName: txt_SingleItemNameNewOrder.Text,
        Quantity: Value(txtQuantityNewOrder.Text),
        Unit: ddUnitNewOrder.Selected.Value
    }
);

// Remove item (Delete button in gallery)
Remove(colOrderItems, ThisItem);
```

How the materials list works:

- Clear() clears the materials list before starting a new order

- Collect adds a new line item with user-entered data
- Value() converts quantity text into a number
- Remove deletes the selected line item from the list (button in the row)

5. Installation Requirements

Prerequisites for deployment:

5.1. Licenses

- Microsoft 365 E3/E5 or Business Premium
- Power Apps per-user or per-app license
- Power Automate Premium (for Dataverse connector)
- Dataverse database capacity

5.2. Environment Configuration

1. Create a Production environment with Dataverse
2. Import solution (MaterialOrder_1_0_0_9.zip)
3. Configure connection references:
   - Dataverse connector
   - Teams connector
4. Update Teams channel IDs in flows (see §3.1.1)
5. Enable flows: NewOrderAdded, OrderResubmitted
6. Add the app to Teams: App Studio → Upload custom app
7. Populate CompanyUser records with user emails and roles

5.3. Security Configuration

The system controls data access depending on the user role:

- **Employee role:** can see only their own orders
- **Supplier role:** can see and edit orders assigned to them
- **Manager role:** has full access to all orders and projects

Security is implemented by:

- Canvas App filtering data by role (see §4.2.1)
- Dataverse controlling access at the database level
- Record owner can share records with other users

6. Usage Example

Full order process – from submission to fulfillment:

6.1. Scenario: Construction Materials Order

Context:

- **Project:** Warehouse Renovation (Legnica)
- **Employee:** Jan Kowalski (Supervisor)
- **Manager:** Anna Nowak (Manager)
- **Supplier:** BuildMat Sp. z o.o. (Supplier)

6.2. Step-by-Step Process

Step 1: Creating the Order

Jan opens Material Order in Teams → New Order → selects:

- Project: Warehouse Renovation
- Supplier: BuildMat Sp. z o.o.
- Priority: High
- Items: 50kg Portland cement, 200 pcs red brick

He clicks “Send for approval” → status changes to Submitted.

Step 2: Approval Notification

Flow NewOrderAdded triggers → checks Project.RequiresApproval = true → sends Adaptive Card to Teams channel.

Anna receives a Teams notification:
“New order ORD-00042 is waiting for your approval”

Step 3: Manager Review

Anna opens the Adaptive Card → reviews order details → clicks “Approve” → the flow updates:

Status: Approved
ApprovedBy: Anna Nowak
ApprovedOn: 2025-02-01T14:23:00Z

Step 4: Supplier Fulfillment

BuildMat user opens Material Order in Teams → sees ORD-00042 in assigned orders → changes status to “In Progress” → marks items as purchased → changes status to “Completed”.

6.3. Alternative Path: Changes Requested

If Anna clicks “Request Changes” and enters:

Status: Draft
RequestChanges: Delivery address confirmation required

Jan sees the order returned to Drafts with feedback → updates delivery address → resubmits → Flow OrderResubmitted triggers → Anna receives a new approval card.

7. Technical Support

For implementation or customization:

Developer	Patryk Lyszkowski
GitHub	github.com/patryk-lyszkowski
LinkedIn	linkedin.com/in/patryk-lyszkowski