Certainly. I'll provide more detailed written instructions to accompany the code examples. These instructions will guide your interface designer through the process of implementing the required features.

1. Implementing PostgreSQL Views in MS Access:

a. PostgreSQL View Creation:

First, connect to your PostgreSQL database using a tool like pgAdmin or psql. Execute the provided SQL script to create the `mrl\_fulfillment\_view`. This view combines data from MRL\_line\_items, fulfillment\_items, and statuses tables.

b. MS Access Function for View Retrieval:

In your MS Access VBA environment, create a new module named "modPostgresViews". Within this module, implement the `GetMRLFulfillmentView` function. This function uses the global connection (g\_conn) to execute a SELECT query on the PostgreSQL view and returns an ADODB.Recordset.

2. Creating Parent-Child Form for MRL and Fulfillment Records:

a. Parent Form (MRL Line Items):

- Create a new form in MS Access Design View.

- Add text boxes for each MRL line item field (jcn, twcode, nomenclature, etc.).

- Set these controls to be read-only for regular users.

- Name this form "frmMRLLineItem".

b. Child Form (Fulfillment Records):

- Create another new form in Design View.

- Add text boxes for fulfillment record fields (status, edd\_to\_ches, rcd\_v2x\_date, etc.).

- Allow these controls to be editable.

- Name this form "frmFulfillmentRecords".

c. Combining Parent and Child Forms:

- Open "frmMRLLineItem" in Design View.

- From the Controls section, add a Subform/Subreport control.

- Set its Source Object to "frmFulfillmentRecords".

- In the Property Sheet for the subform control, set:

- Link Master Fields: order\_line\_item\_id

- Link Child Fields: order\_line\_item\_id

d. Loading Data:

- In the VBA editor, open the code behind "frmMRLLineItem".

- Implement the Form\_Load event as shown in the provided code.

- This code retrieves data from the PostgreSQL view and sets up the link between parent and child forms.

3. Implementing Fulfillment Record Search:

a. Search Form Creation:

- Create a new form named "frmSearch".

- Add text boxes for search criteria (e.g., txtJCN, txtTWCode).

- Add a command button named "cmdSearch".

b. Search Functionality:

- In the VBA editor, open the code behind "frmSearch".

- Implement the cmdSearch\_Click event as shown in the provided code.

- This function constructs a SQL query based on the entered search criteria and executes it using the global connection.

4. Implementing Fulfillment Record Update:

a. Update Form Creation:

- Create a new form named "frmUpdateFulfillment" or add update functionality to "frmFulfillmentRecords".

- Ensure there are editable text boxes for fulfillment record fields.

- Add a command button named "cmdUpdate".

b. Update Functionality:

- In the VBA editor, open the code behind the update form.

- Implement the cmdUpdate\_Click event as shown in the provided code.

- This function constructs an UPDATE SQL statement based on the changed fields and executes it using the global connection.

5. Implementing Full History View for Fulfillment Records:

a. PostgreSQL Function Creation:

Connect to your PostgreSQL database and execute the provided SQL script to create the `get\_fulfillment\_history` function. This function retrieves the full history of a fulfillment record, including audit trail, comments, and inquiry status changes.

b. VBA Function for History Retrieval:

In your MS Access VBA environment, create a new module named "modFulfillmentHistory". Implement the `GetFulfillmentHistory` function as shown in the provided code. This function calls the PostgreSQL function and returns the results as an ADODB.Recordset.

c. History View Form Creation:

- Create a new form named "frmFulfillmentHistory".

- Add a subform control to display the history records.

- In the form's Load event, implement the code to retrieve and display the history for a specific fulfillment item.

6. Using the DSN-less Global Connectivity Architecture:

a. Connection Initialization:

- Ensure that the `InitializePostgresConnection` function in the "modGlobalConnection" module is called when the application starts. This is typically done in the AutoExec macro or a startup form.

b. Using the Global Connection:

- Before performing any database operation, always check if the connection is initialized:

```vba

If g\_conn Is Nothing Then

InitializePostgresConnection

End If

```

- Use the `g\_conn` object for all database operations, as shown in the provided code examples.

c. Error Handling:

- Implement error handling in all functions that interact with the database.

- Use On Error GoTo statements to catch and handle any connection or query execution errors.

General Tips for the Interface Designer:

1. Consistent Naming: Use a consistent naming convention for all forms, controls, and modules.

2. Comments: Add comments to the code to explain complex logic or non-obvious operations.

3. Error Handling: Implement proper error handling throughout the application.

4. Testing: Thoroughly test each feature with various scenarios, including edge cases.

5. User Feedback: Provide clear feedback to users through message boxes or status updates.

6. Performance: For large datasets, consider implementing pagination or limiting the number of records returned in a single query.

By following these detailed instructions and adapting the provided code examples, your interface designer should be able to implement the required features using the DSN-less global connectivity architecture.