Functionality Flow

Data from some other platform like SAP loaded into SnowFlake

From Snowflake, we get two sets of data to mendix. Data can be anything, so far, no idea

One set of data (order list) is IST- No edition can be done there. Just to view

Other set of data (order list), we can edit the sort of value of the orders based on weeks (which data to be delivered first, based on priority of the customers)

At present we are generating raw data in csv and importing that in mendix and converting that to json from there using import mapping in mendix and committing it in db – performance issue

**Capacity-Table**

Capacity table is the table that should show the calendar from the start date of input end ends with +8 days of last date of input.- false Orders

Implemented:

Calendar render is static for the given inputs .& every time when team changes , capacity is calculated based on teams.

Calendar starts from data -done

Ends after add 6 weeks

Check with dropdown tooo.

**Button on right corner of the page**

From UI side

Button on top of overview page  
Three slider on pop page (Criteria 1, criteria 2, Criteria 3). In domain model, criteria name and unit(percentage). Slider values between 0 to 100.

Run algorithm button and Cancel button

Functionality:

When button clicked some api in background runs , In UI button load render ,api in background hits generates transaction id & using the id as input hits other api & returns json. - this functionality retrieves false order from api & stores in db

Other Learnings

InputStream / bufferReader

BufferedReader reads a couple of characters from the Input Stream and stores them in a buffer.

InputStreamReader reads only one character from the input stream and the remaining characters still remain in the streams hence There is no buffer in this case.

Xpath: with or /and

Subconstraint-and

XPath<Orders> xpathBuilder = XPath.*create*(getContext(), Orders.**class**);

xpathBuilder.eq(Orders.MemberNames.***IstSituation***,**this**.istSituation);

XpathBuilder.subconstraint(Orders.MemberNames.***Orders\_ProductTeam***,ProductTeam.***entityName***)

.eq(ProductTeam.MemberNames.***Name***,productTeamList.get(i).getName());

Obj retrieval:

ProductTeam productTeam = XPath.*create*(context, ProductTeam.**class**)

.eq(ProductTeam.MemberNames.***Name***, “TeamA”).first();

Xpath to delete from entity

XPath.*create*(getContext(), AssetCurrentAggregation.**class**).deleteAll();

List:

List<Orders> OrderList = XPath.*create*(context, Orders.**class**). eq(Orders.MemberNames.***IstSituation***, bool).all()

List<IMendixObject> list = new ArrayList<IMendixObject>();

Orders order = **new** Orders(getContext());

order.setMaterialNo(matNrObject.getString(key));

Orderlist.add(order.getMendixObject());

React

UseState:

useState is a React Hook that lets you add a [state variable](https://react.dev/learn/state-a-components-memory) to your component.

State variable?

Components often need to change what’s on the screen as a result of an interaction. In React, this kind of component-specific memory is called *state*.

To update a component with new data, two things need to happen:

1. **Retain** the data between renders.
2. **Trigger** React to render the component with new data (re-rendering).

The [useState](https://react.dev/reference/react/useState) Hook provides those two things:

1. A **state variable** to retain the data between renders.
2. A **state setter function** to update the variable and trigger React to render the component again.

Microflow running in background can be achieved in mendix by:

1. ExecuteMicroflowInBackground
2. MicroFlow Timer
3. Task Queue

JavaScript

-----------

String methods

[String length](https://www.w3schools.com/js/js_string_methods.asp#mark_length)[String charAt()](https://www.w3schools.com/js/js_string_methods.asp#mark_charat)[String charCodeAt()](https://www.w3schools.com/js/js_string_methods.asp#mark_charcodeat)[String at()](https://www.w3schools.com/js/js_string_methods.asp#mark_at)[String [ ]](https://www.w3schools.com/js/js_string_methods.asp#mark_propertyaccess)[String slice()](https://www.w3schools.com/js/js_string_methods.asp#mark_slice)[String substring()](https://www.w3schools.com/js/js_string_methods.asp#mark_substring)[String substr()](https://www.w3schools.com/js/js_string_methods.asp#mark_substr)

[String toUpperCase()](https://www.w3schools.com/js/js_string_methods.asp#mark_touppercase)[String toLowerCase()](https://www.w3schools.com/js/js_string_methods.asp#mark_tolowercase)[String concat()](https://www.w3schools.com/js/js_string_methods.asp#mark_concat)[String trim()](https://www.w3schools.com/js/js_string_methods.asp#mark_trim)[String trimStart()](https://www.w3schools.com/js/js_string_methods.asp#mark_trimstart)[String trimEnd()](https://www.w3schools.com/js/js_string_methods.asp#mark_trimend)[String padStart()](https://www.w3schools.com/js/js_string_methods.asp#mark_padstart)[String padEnd()](https://www.w3schools.com/js/js_string_methods.asp#mark_padend)[String repeat()](https://www.w3schools.com/js/js_string_methods.asp#mark_repeat)[String replace()](https://www.w3schools.com/js/js_string_methods.asp#mark_replace)[String replaceAll()](https://www.w3schools.com/js/js_string_methods.asp#mark_replaceall)[String split()](https://www.w3schools.com/js/js_string_methods.asp#mark_split)

Search methods

[String indexOf()](https://www.w3schools.com/js/js_string_search.asp#mark_indexof)[String lastIndexOf()](https://www.w3schools.com/js/js_string_search.asp#mark_lastindexof)[String search()](https://www.w3schools.com/js/js_string_search.asp#mark_search)

[String match()](https://www.w3schools.com/js/js_string_search.asp#mark_match)[String matchAll()](https://www.w3schools.com/js/js_string_search.asp#mark_matchall)[String includes()](https://www.w3schools.com/js/js_string_search.asp#mark_includes)[String startsWith()](https://www.w3schools.com/js/js_string_search.asp#mark_startswith)[String endsWith()](https://www.w3schools.com/js/js_string_search.asp#mark_endswith)