A picture containing sitting, monitor, plate, clock

Description automatically generated

**Functional Specification Document**

**Visual Investigator**

(Alert History and KYC)

Version 1

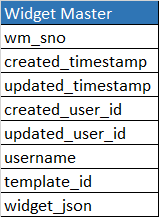
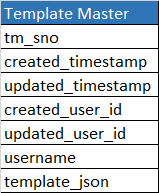
**Alert History**

Alert History visual will consist of alerts generated and closed alerts for the customer in the given time-frame. All the components related to this module are covered here in depth:

1. Assumption and Exceptions
2. Data Model
3. Sync Job
4. Template Creation
5. Template Rendering

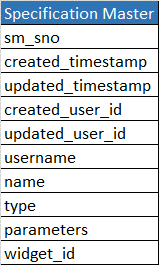
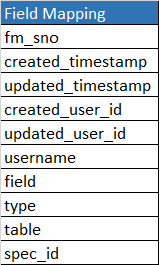
**Assumptions & Exceptions:**1. The Database used in Presentation Layer is PostgreSQL  
2. Use of Scheduled Stored procedure for syncing data in DB from Mart.  
3. No. of widgets and specification are static or either configurable from DB

**Data Model:**

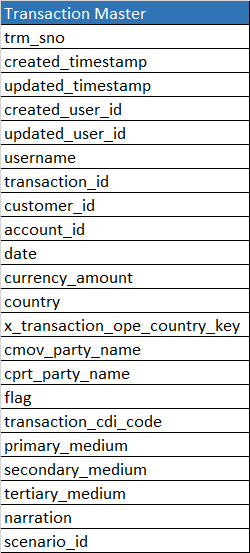


1:n

1:n

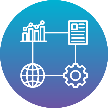


1:n



**Sync Job:**

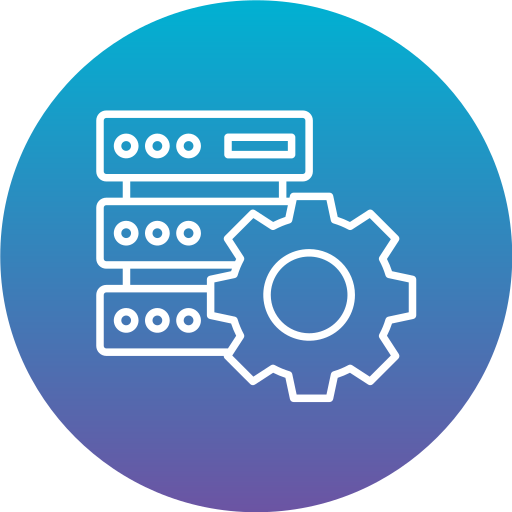
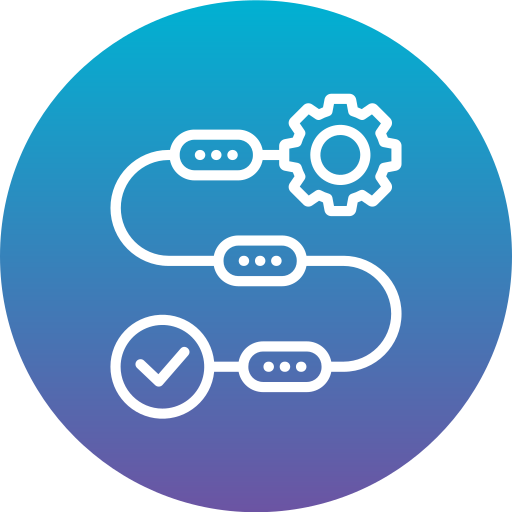
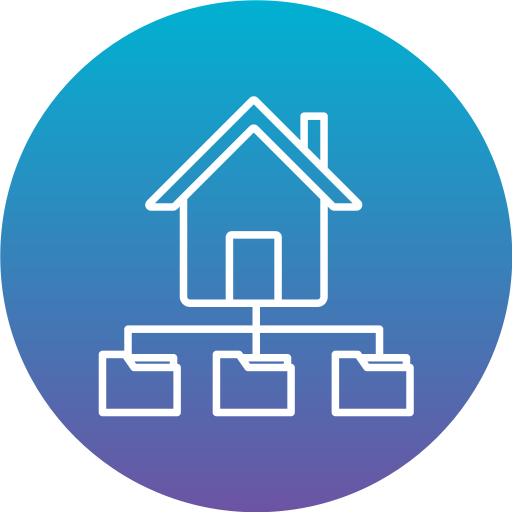
The role of this scheduled task is to fetch required Data from Data mart and save it in our presentation layer after performing ETL over it. Once this operation is done we get refined data stored, which can now be utilised for performing visualization. The scheduling process can be done either by creating an API or using a SQL script and then scheduling it as cronjob.



Data Mart

Scheduled ETL

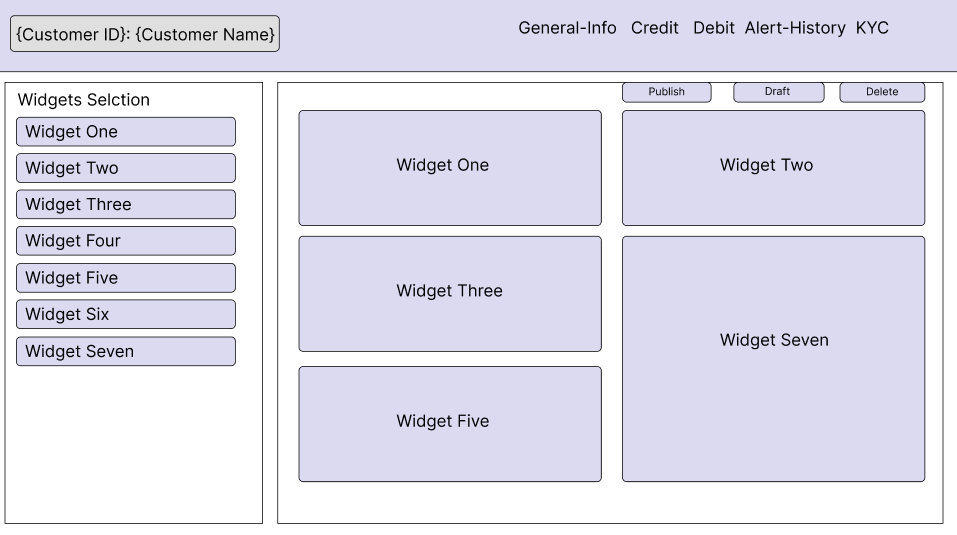
Presentation DB

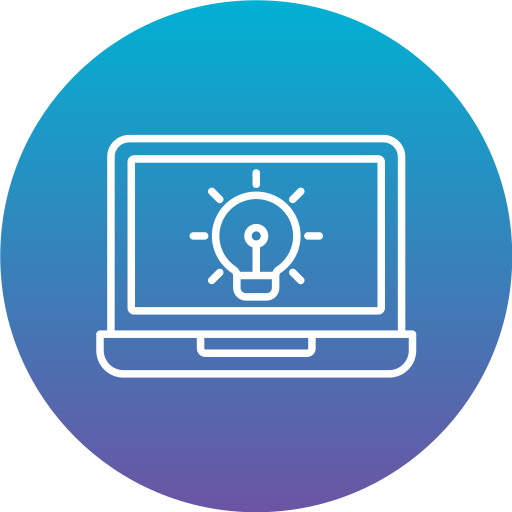


{ proc sql;  
……………  
…………. }

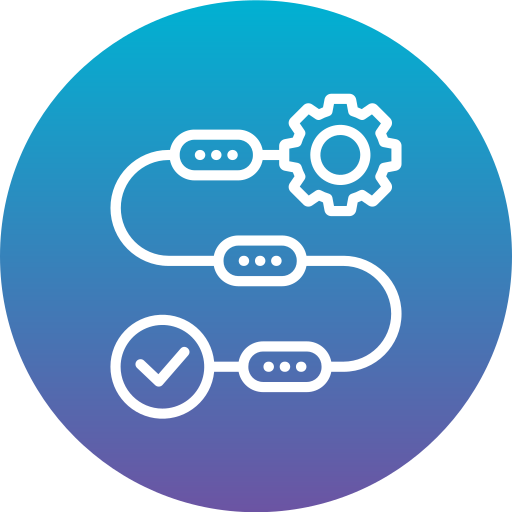
The approach to create Sync Job will be done using Stored Process (SP). The GET API will be created on top of this SP to call it and it can be scheduled to make periodic calls.

**Template Creation:**Template creation is very unique feature of this module. User can drag & drop various widgets on the blank template and publish it to visualise desired results. The List of widgets and specification can vary from page to page.

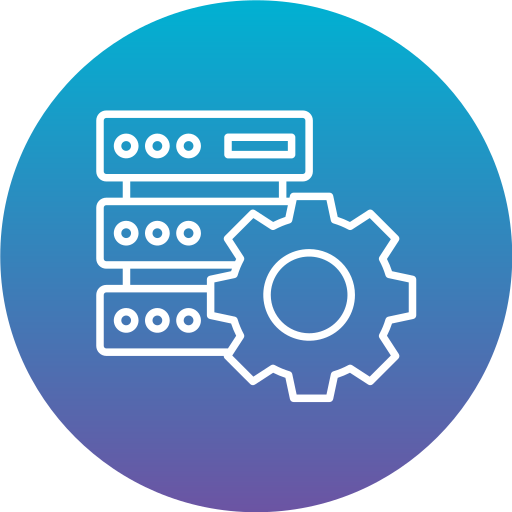
****



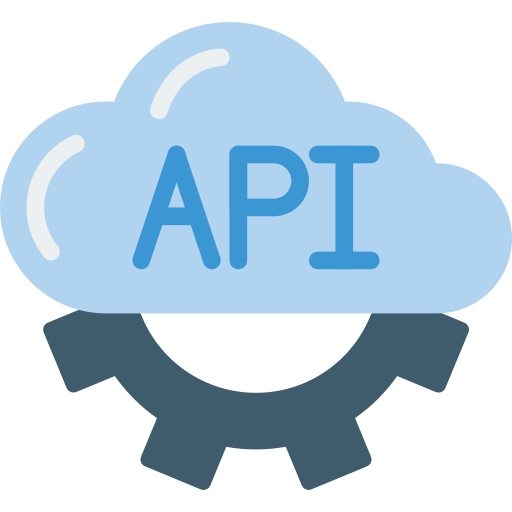
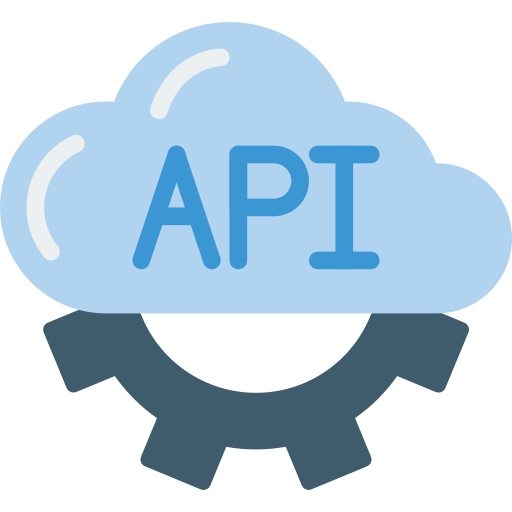
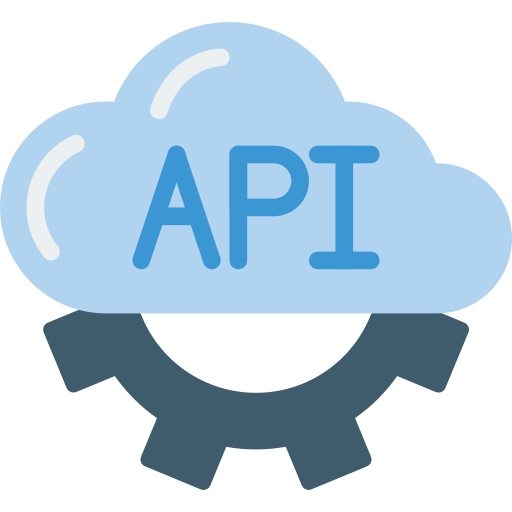
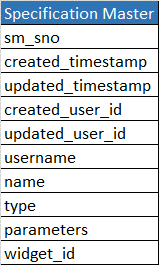
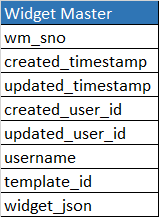
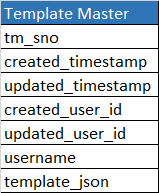
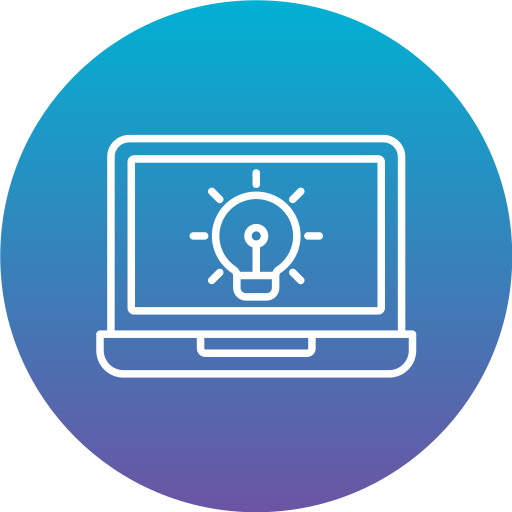
Widgets



Specification



Template Page



Template is the very fundamental component of the module. It acts as a superset to widget. Templates are owned by users and are saved in table accordingly

* Template is JSON file consisting of formatted structure.
* Template can have multiple widgets which user can select from a fixed list (configurable in Widget Master). It’s a piece of JSON code stored in Database in Widget Master.
* Each widget in turn have a set of specification, which helps us in forming request structure for REST APIs. Specification are linked with fields in Field Mapping Table.
* Template can be published, updated and deleted by the owner user.

**Template Rendering:**

This the last stage where saved template will be rendered to create visualisation for the selected customer. Rendering will happen in the following steps:

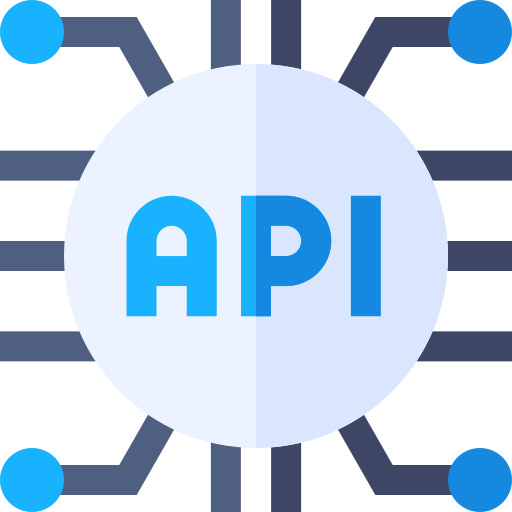
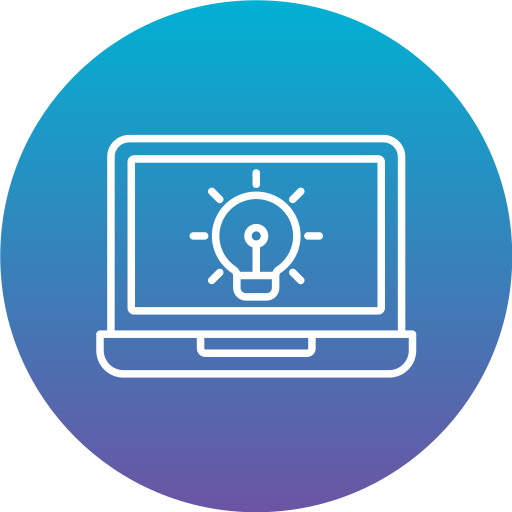
* UI will fetch the selected template and based on its widgets and specification configuration it will further make request to API server and fetch all data.
* User can change the date range and based on that the data will be updated on UI.

**Rendering Cases:**Rendering cases are based on the existing layout received. For History we have 2 widgets and 4 specifications listed below:

**Widget 01: Alerts**  
Specifications: Alerts count by month, Alerts count by week, Alerts count by day

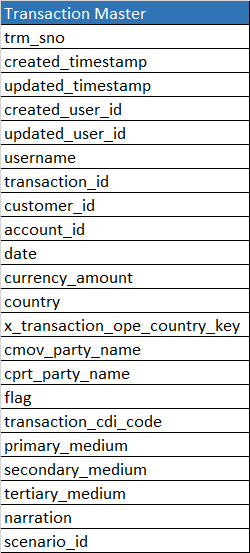
Case: 01 - Alerts count by month  
In this case the widget will fetch the corresponding fields of specification and apply month based filtering on the data and return the count of transaction where alert flag is true in transaction\_master corresponding to customer\_id.

Template Page



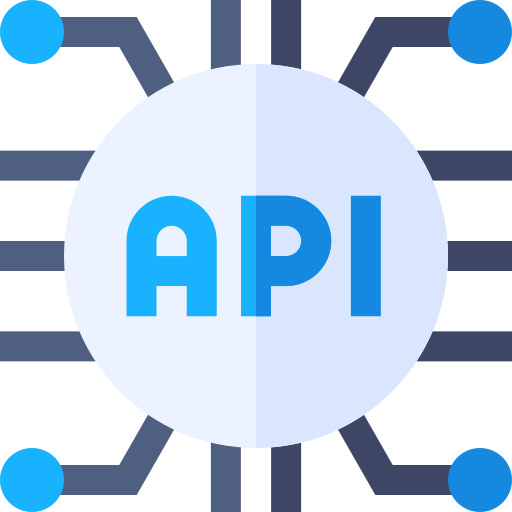
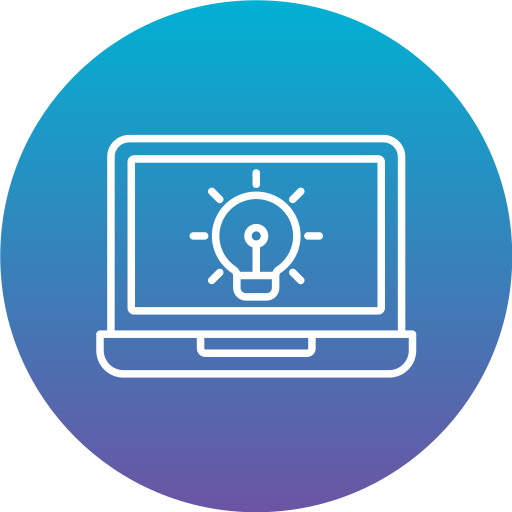
API Server

If(flag==true) {  
 List = SELECT \* from transaction\_master WHERE (date.month IN dto.date.range) AND created\_user\_id == session.user\_id)  
}



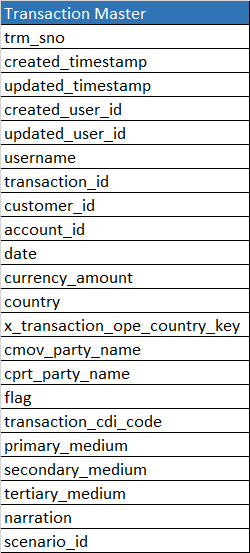
Case: 02 - Alerts count by week  
In this case the widget will fetch the corresponding fields of specification and apply week based filtering on the data and return the count of transaction where alert flag is true in transaction\_master corresponding to customer\_id.

Template Page



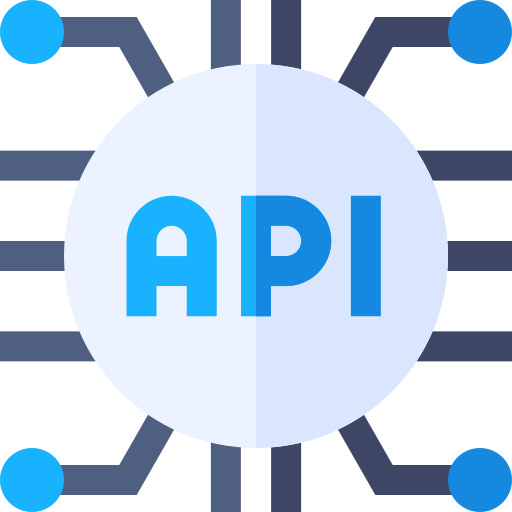
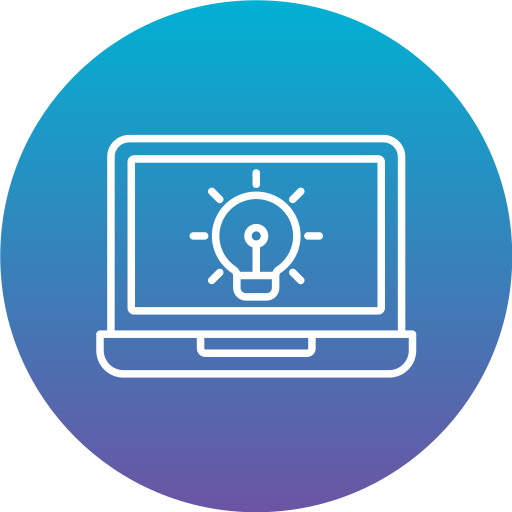
API Server

If(flag==true) {  
 List = SELECT \* from transaction\_master WHERE (date.day >= dto.date-7) AND (created\_user\_id == session.user\_id)  
}



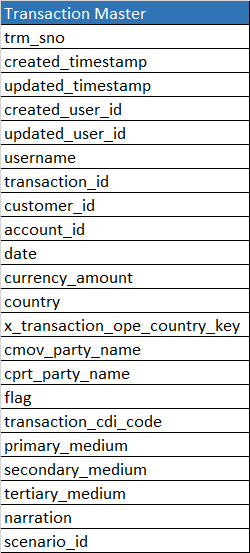
Case: 03 - Alerts count by day  
In this case the widget will fetch the corresponding fields of specification and apply day based filtering on the data and return the count of transaction where alert flag is true in transaction\_master corresponding to customer\_id.

Template Page



API Server

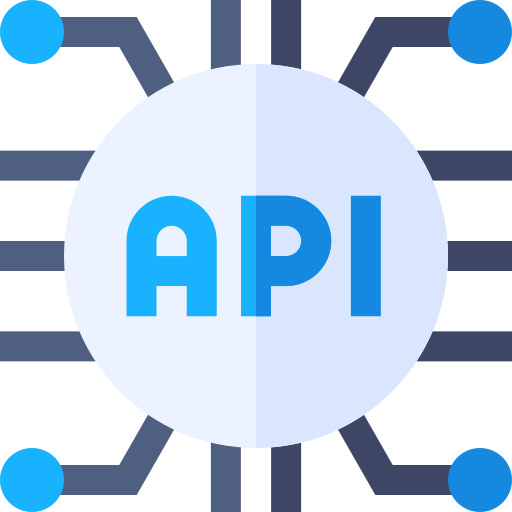
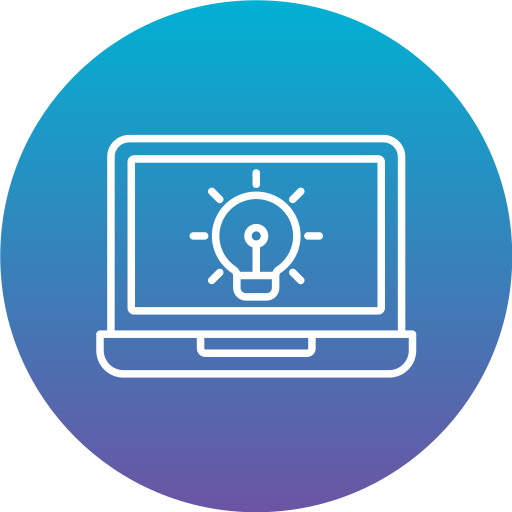
If(flag==true) {  
 List = SELECT \* from transaction\_master WHERE (date.date == dto.date) AND (created\_user\_id == session.user\_id)  
}



**Widget 02: Narration**  
Specifications: Last closed alerts

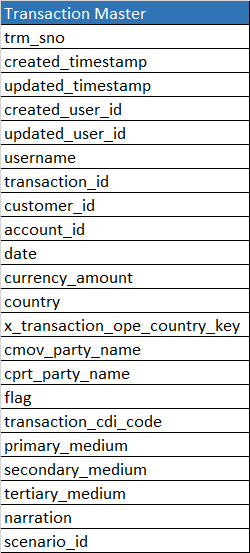
The alerts closed on all previous entries in the give date range will be fetched along with the scenario\_id and narration. Given that alert flag is false.

Template Page



API Server

If(flag==false) {  
 List = SELECT \* from transaction\_master WHERE (date.date == dto.date.range) AND   
(created\_user\_id == session.user\_id)  
}



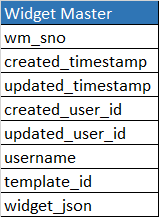
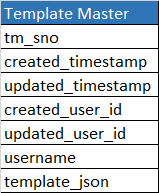
**KYC**

KYC visual will consist of General information and Risk Scorecard for the customer. All the components related to this module are covered here in depth:

1. Assumption and Exceptions
2. Data Model
3. Sync Job
4. Template Creation
5. Template Rendering

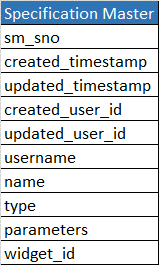
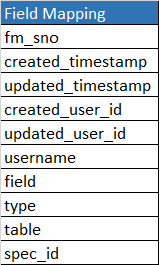
**Assumptions & Exceptions:**1. The Database used in Presentation Layer is PostgreSQL  
2. Use of Scheduled Stored procedure for syncing data in DB from Mart.  
3. No. of widgets and specification are static or either configurable from DB

**Data Model:**

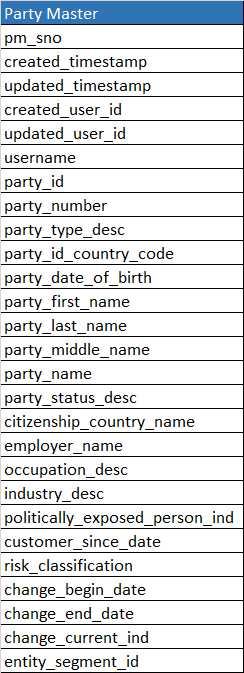


1:n

1:n

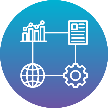


1:n



**Sync Job: (Same as Alert History)**

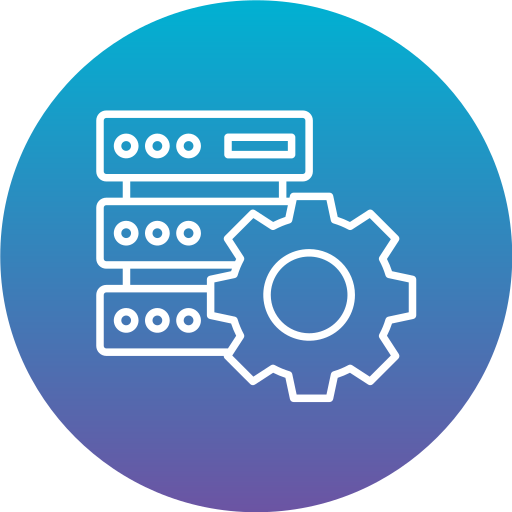
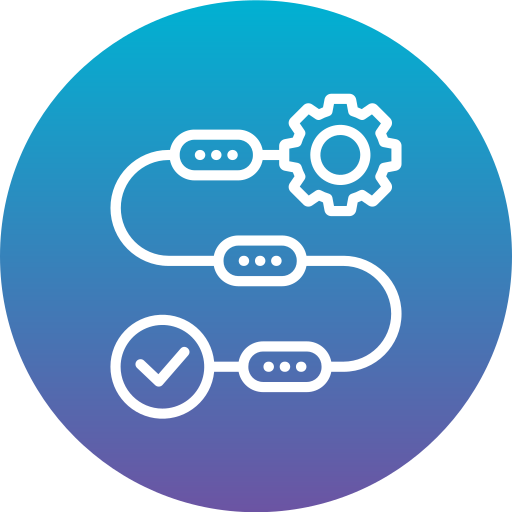
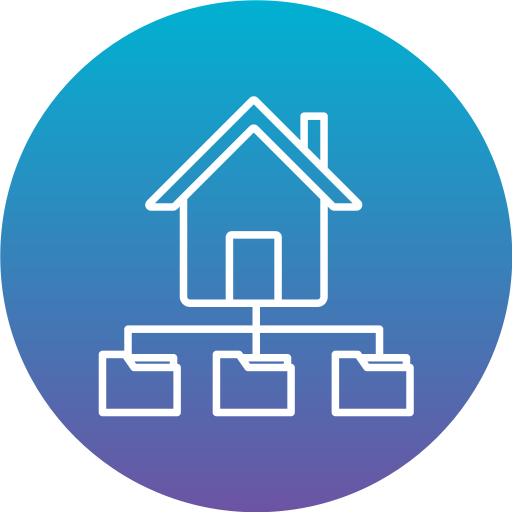
The role of this scheduled task is to fetch required Data from Data mart and save it in our presentation layer after performing ETL over it. Once this operation is done we get refined data stored, which can now be utilised for performing visualization. The scheduling process can be done either by creating an API or using a SQL script and then scheduling it as cronjob.



Data Mart

Scheduled ETL

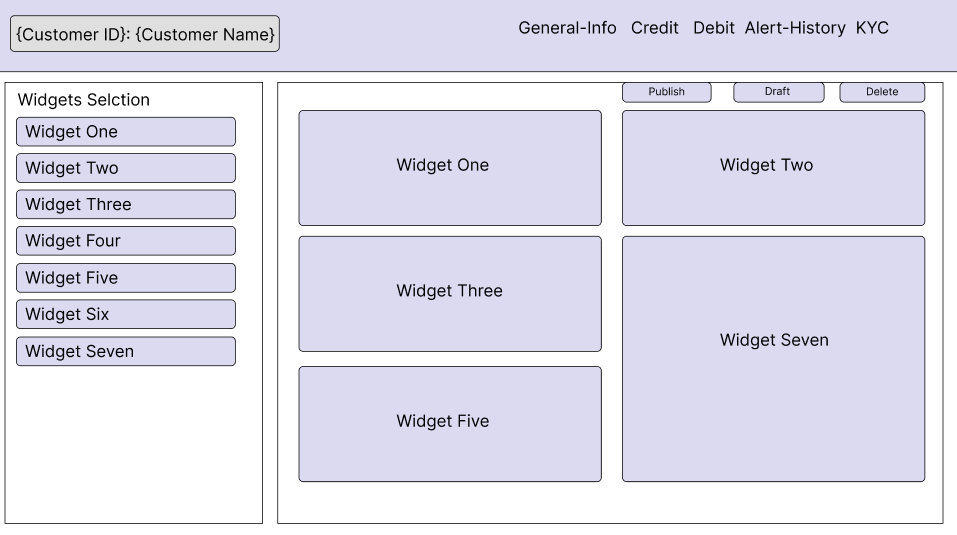
Presentation DB

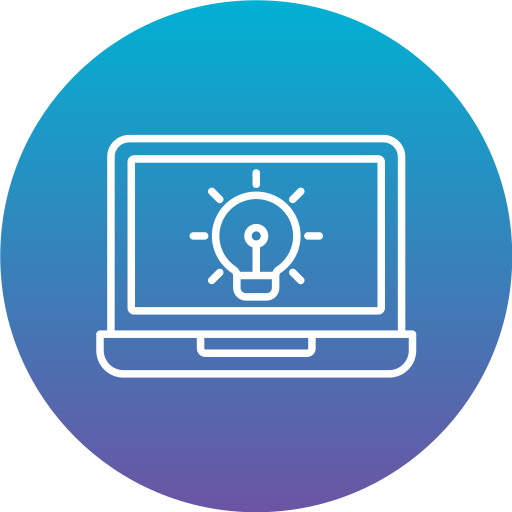


{ proc sql;  
……………  
…………. }

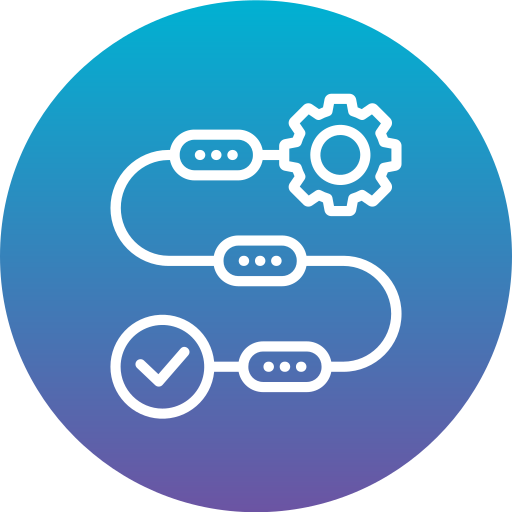
The approach to create Sync Job will be done using Stored Process (SP). The GET API will be created on top of this SP to call it and it can be scheduled to make periodic calls.

**Template Creation: (Same as Alert History)**Template creation is very unique feature of this module. User can drag & drop various widgets on the blank template and publish it to visualise desired results. The List of widgets and specification can vary from page to page.

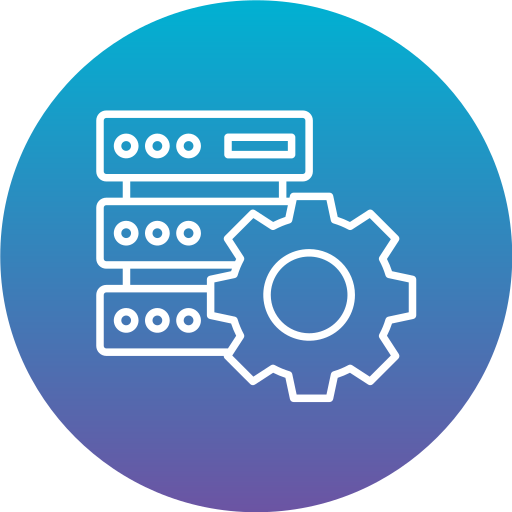
****



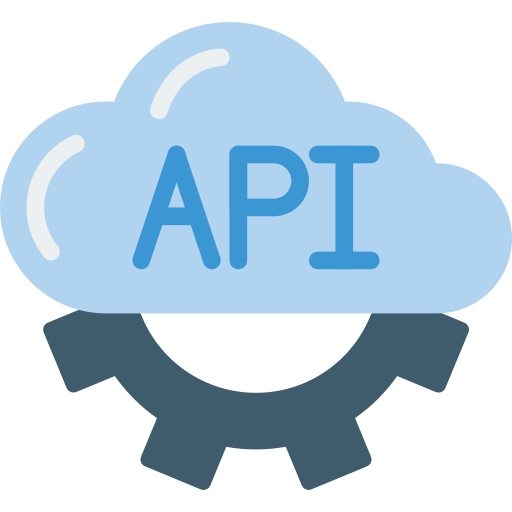
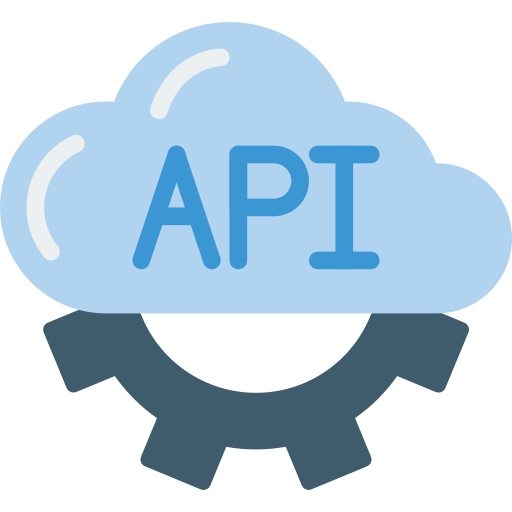
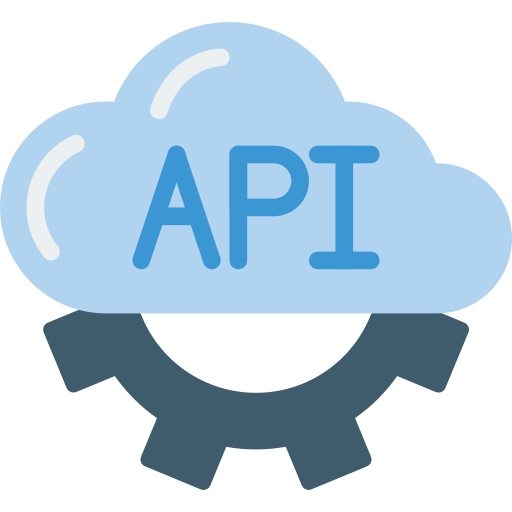
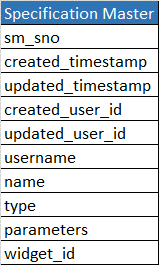
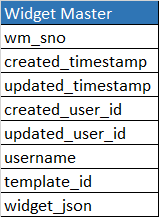
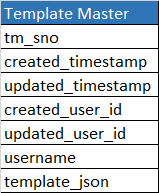
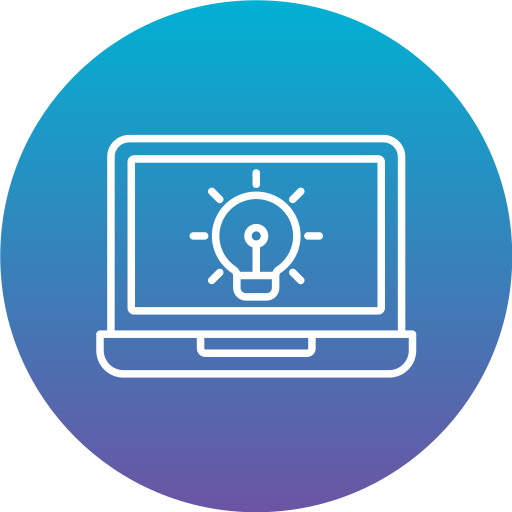
Widgets



Specification



Template Page



Template is the very fundamental component of the module. It acts as a superset to widget. Templates are owned by users and are saved in table accordingly.

* Template is JSON file consisting of formatted structure.
* Template can have multiple widgets which user can select from a fixed list (configurable in Widget Master). It’s a piece of JSON code stored in Database in Widget Master.
* Each widget in turn have a set of specification, which helps us in forming request structure for REST APIs. Specification are linked with fields in Field Mapping Table.
* Template can be published, updated and deleted by the owner user.

**Template Rendering:**

This the last stage where saved template will be rendered to create visualisation for the selected customer. Rendering will happen in the following steps:

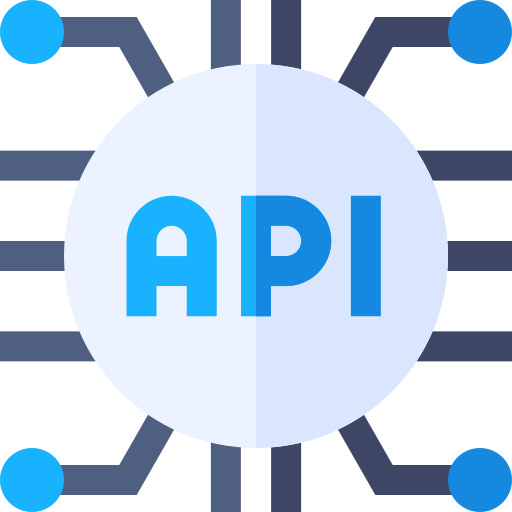
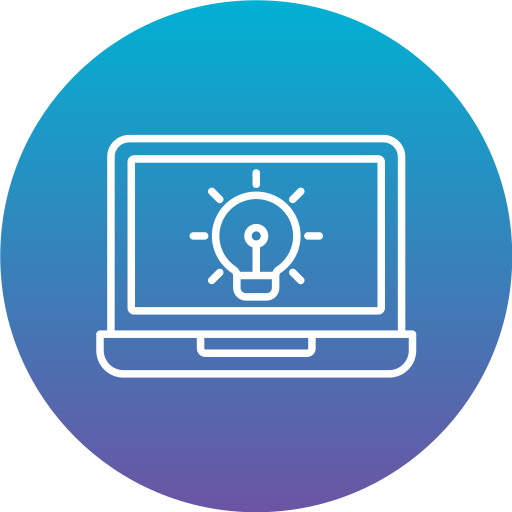
* UI will fetch the selected template and based on its widgets and specification configuration it will further make request to API server and fetch all data.
* User can change the date range and based on that the data will be updated on UI.

**Rendering Cases:**Rendering cases are based on the existing layout received. For History we have 2 widgets and 4 specifications listed below:

**Widget 01: General Information**  
Specifications: Customer Information

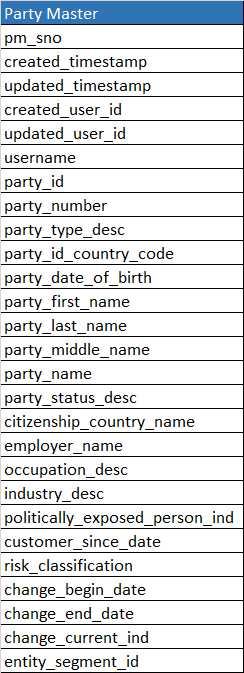
Case: 01 – Customer Information  
In this case the widget will fetch the corresponding fields of specification party\_master based on the customer\_id.

Template Page



API Server

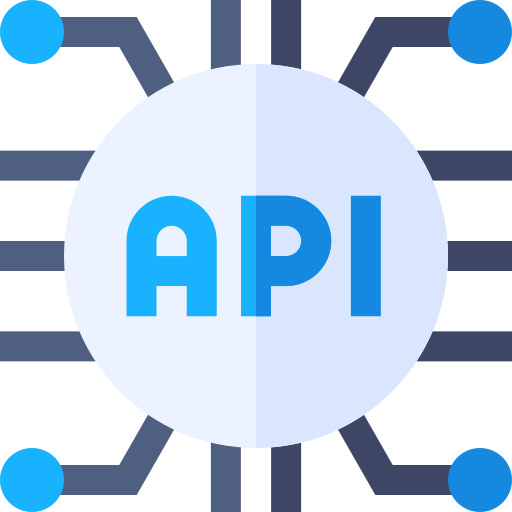
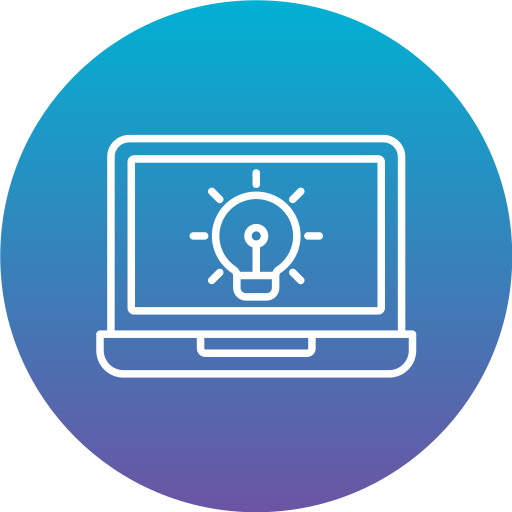
{  
 List = SELECT \* from party\_master WHERE (party\_id == dto.party\_id)  
}



**Widget 02: Score Card**  
Specifications: Score card

The generated risk\_classification will be fetched from the party\_master based on the customer\_id provided. The classification is generated on the basis of CDD rules performed.

Template Page



API Server

{  
 List = SELECT \* from party\_master WHERE (party\_id == dto.party\_id)  
}

