**API Specification Document**

(Component Management System)

| **Version** | **Date** | **Author** | **Description** |
| --- | --- | --- | --- |
| 1.0 | 08/07/2024 | Ravi R V | Initial draft |

# INDEX

1. GetComponents

Request

Response

2.AddComponents

Request

Response

3. UpdateComponents

Request

Response

4. DeleteComponents

Request

Response

Conventions

Status Codes

Methods

## 1. Get Components

Get the components from the database.

## **Request**

| **Method** | **URL** |
| --- | --- |
| **GET** | https://localhost:7230/api/Components/GetComponents |

## **Response**

| **Status** | **Response** |
| --- | --- |
| **200** | Response will be an object containing the list of the components which are present in the database. Each component will be having the structure like  {  "serialNo": int,  "manufacturerPartNo": "string",  "componentType": "string",  "packageSize": "string",  "qtyAvailable": int,  "entryDate": "DateTime",  "binNo": "string",  "rackNo": "string",  "projectUsed": "string"  }  **An Example response will be like this,**  [  {  "serialNo": 1,  "manufacturerPartNo": "RDT8274",  "componentType": "GH",  "packageSize": "3",  "qtyAvailable": 100,  "entryDate": "2024-06-13T18:27:50.83",  "binNo": "4Bin",  "rackNo": "14R",  "projectUsed": "REM"  },  {  "serialNo": 2,  "manufacturerPartNo": "RT119",  "componentType": "G",  "packageSize": "4L",  "qtyAvailable": 56,  "entryDate": "2024-06-13T18:29:05.727",  "binNo": "32e",  "rackNo": "A3",  "projectUsed": "RC"  }  ] |
| **500** | “error” : Internal server error |

## 2. AddComponents

This method is used to add the new components to the database. This will be done at either front end or any tool like postman or swagger.UI and it should replicate in the database whenever the user adds the new component.

## **Request**

| **Method** | **URL** |
| --- | --- |
| **POST** | https://localhost:7230/api/Components/AddComponents |

The example Request body of the application/json will be like this

{

"serialNo": 0,

"manufacturerPartNo": "string",

"componentType": "string",

"packageSize": "string",

"qtyAvailable": 0,

"entryDate": "2024-07-08T11:41:15.517Z",

"binNo": "string",

"rackNo": "string",

"projectUsed": "string"

}

## **Response**

| **Status** | **Response** |
| --- | --- |
| 201 | The new component with it’s data will be added to the database. An example {  "serialNo": 10,  "manufacturerPartNo": "IUD34",  "componentType": "ELc",  "packageSize": "Medium",  "qtyAvailable": 40,  "entryDate": "2024-07-08T11:41:15.517Z",  "binNo": "14B",  "rackNo": "3R",  "projectUsed": "CMS"  } |
| 400 | Bad Request,   1. One or more validation errors occurred. 2. A non-empty request body is required.   Example error while submitting with the empty fields.  "errors": {  "BinNo": [  "The BinNo field is required."  ],  "RackNo": [  "The RackNo field is required."  ],  "PackageSize": [  "The PackageSize field is required."  ],  "ProjectUsed": [  "The ProjectUsed field is required."  ],  "ComponentType": [  "The ComponentType field is required."  ],  "ManufacturerPartNo": [  "The ManufacturerPartNo field is required."  ] |
| 500 | Internal server error |

## 3. Update Components

This method is used to edit the already existing components as for the convenience of the user requirements or if the user has committed any mistakes while adding the components. This will be done at either front end or any tool like postman or swagger.UI.

This method will update the components data in the database whenever the user makes any changes to the current data.

## **Request**

| **Method** | **URL** |
| --- | --- |
| **PUT** | /api/Components/UpdateComponents |

## **Response**

| **Status** | **Response** |
| --- | --- |
| **404** | Error: response status is 404 Component not found is returned since the user has not provided the correct serialNo.  Request body: {  "serialNo": 0,  "manufacturerPartNo": "string",  "componentType": "string",  "packageSize": "string",  "qtyAvailable": 0,  "entryDate": "2024-07-09T12:01:48.702Z",  "binNo": "string",  "rackNo": "string",  "projectUsed": "string"  } |
| **200** | Response body:  1  Returns the number of rows affected, which are succesfull. Example for successfull data for update component, {  "serialNo": 1,  "manufacturerPartNo": "RDT8274",  "componentType": "GH",  "packageSize": "3",  "qtyAvailable": 100,  "entryDate": "2024-06-13T18:27:50.83",  "binNo": "4Bin",  "rackNo": "14R",  "projectUsed": "CMS"  }  Here the projectUsed value has been changed from REM to CMS and hence the result is produced as 1(the no. of rows affected). The component with the serialNo: 1 is updated. |

## 4. Delete Components

User can delete the components which are not required, It will return the number of rows affected in the database if the deletion is successful. The user has to provide the serialNo as the parameter since it is the unique key in the table for each component.

## **Request**

| **Method** | **URL** |
| --- | --- |
| **PUT** | /api/Components/DeleteComponent/{serialNo} |

## **Response**

| **Status** | **Response** |
| --- | --- |
| **404** | Request URL : https://localhost:7230/api/Components/DeleteComponent/0  Error: response status is 404  Response body:  Component not found. If the serialNo provided as a parameter is either null or which is not present in the table will return the result as error 400 which is Component not found. |
| **200** | Request Url : https://localhost:7230/api/Components/DeleteComponent/21  Response body:  1  Returns the number of rows affected in the table, which are successful in deleting the component according to their serialNo. |

## 

Glossary

## **Conventions**

* **Client** - Client application.
* **Status** - HTTP status code of response.
* All the possible responses are listed under ‘Responses’ for each method. Only one of them is issued per request server.
* All response are in JSON format.
* All request parameters are mandatory unless explicitly marked as [optional]

## **Status Codes**

All status codes are standard HTTP status codes. The below ones are used in this API.

| **Status Code** | **Description** |
| --- | --- |
| 200 | OK |
| 201 | Created |
| 202 | Accepted (Request accepted, and queued for execution) |
| 400 | Bad request |
| 404 | Not found |
| 500 | Internal Server Error |