

iRest: Rest API Simulator

Product Requiremnt

document

Table of Contents

[Goals & Objectives 2](#_Toc64646124)

[User Mock creation Journey 2](#_Toc64646125)

[User Mock Updation Journey 2](#_Toc64646126)

[Common Fields 2](#_Toc64646127)

[ Default Type 3](#_Toc64646128)

[ Save Type 4](#_Toc64646129)

[ Existing Type 4](#_Toc64646130)

[Use cases 5](#_Toc64646131)

[Requirements 5](#_Toc64646132)

## Goals & Objectives

iRest should be able to provide the RESTful API Platform which can be plugged to any project with mock data. iRest would provide a way to create any mock service on the air and can be utilized in your new or existing project.

## User MOCK creation Journey

1. User should be start from a Login Screen.
2. After successful login, User would be redirected to Mock List Page where Mock List should have Pagination and Sorting options.
3. The Mock Listing Page should have an option for adding a new mock. Also for individual mock, there has been an option for edit, delete and disabled/enabled the mock.
4. When User Comes to Mock Creation Page, there should be able to add mock option and add the metadata.
5. On submit the mock options and metadata. The use will create on submit button and the data should be stored and then, flow will redirect user to mock list page with updated mock list.

## USER MOCK Updation JOURNEY

1. Follow the steps 1 to 3 from mock creation journey.
2. From Mock Listing Page, User can click on edit button and will be redirect to mock edit page. All the Mock API options and metadata should be pre populated.
3. Endpoint & Project Name field should be non-editable for edit mock journey.
4. Follow step 5 from mock creation journey.

Below are the fields for Mock API options and metadata:

## Common Fields

1. **Base** URL: Base URL of the Mock API
2. **Project Name**: Field should be searchable from among all the user generated project name. Also, there has been option for create the new project name.
3. **Endpoint**: Mock URL end point.

The complete URL should be combination of Base URL, Project Name & Endpoint.

***http://localhost/api/rest*** *+* ***test-project-name*** *+* ***test-endpoint***

The complete Mock Url should be like:

[*http://localhost/api/rest/test-project-name/test-endpoint*](http://localhost/api/rest/test-project-name/test-endpoint)

1. **Type of Mock**: Mock Type could be any of three mentioned below:
   1. **Default**: Return the JSON/Plain-Text Data in Response with certain optional advanced options such as delay, dynamic records, etc.
   2. **Save**: Mock APIs with CRUD Operations such as create, modify, view and delete the records.
   3. **Existing**: A Listing API with all the records stored during the CRUD Operation in 2nd type (Save) of API.
2. **Request method**: The Request method of API call among **GET/ POST/ PUT/ PATCH/ DELETE** http methods.
   1. **Default Mock Type:** A user may retrieve the data via GET or POST method only. In this scenario, only GET/ POST methods are sufficient to fetch the data.
3. **Response Status Code**: The Response http status code from API end. It could be any one of these status codes: **200/ 400/ 401/ 500/ 502**.
4. **Advanced Options**: Optional for user, this contains some optional fields user can choose if required.
   1. **Content Encoding**: ‘UTF-8’. Field should be auto populated for now. We will add more encoding in future.
   2. **Headers**: A combination of key/value pair of inputs. A user can fill any amount of custom headers. There is not limit for the number of headers. These custom headers will be added to the Mock API response.
5. **Mock Name:** Name of mock for identification purpose.
6. **Mock Status Checkbox**: A checkbox for toggle the active state of mock. If checkbox is ticked, mock will be enabled. Otherwise user can update/override the mock status from Mock Listing Page.

The fields below are needed on the basis of mock type. The fields may or may not vary for individual use case.

### Default Type

1. **Content Type:** A list of all the content type which is expected from the Mock API response. Available Content Type for present usage: *Text/Plain, Application/JSON*.
2. **Response Body**: The input field for response body which is expected from the Mock API Response. The input field should contain a valid content type. Validate the data before stores the data.
3. **Upload Response Body**: The upload button is to upload a JSON type response. It is useful in case end user has a large file of JSON type response. Once the user upload the file, all the content from should be populated to the Response Body input field.
4. **Advanced Options**: Optional for user, this contains some optional fields user can choose if required.
   1. **Delay:** A button that toggle delay field. If delay is enable user can set the delay the response. Boundaries for delay: Min: 0 seconds, Max: 60 seconds.
   2. **Dynamic Response:** A button that toggle the delay section. Delay Section consists the field mentioned below.
      1. **Key:** A Key from the response body on which dynamic operation is operated at the time of Mock API response.
      2. **Dropdown for Type of Key Value:** Dropdown consists 2 options, random & specific.
         1. For **Random** type of key value, dummy data can be prefilled from the Mock API end.
         2. For **Specific** type of key value, an input field should be added for key value.
      3. **Dropdown for quantity of dynamic records:** Dropdown consists 2 options, count & size.
         1. For **Count** Option, an input field should be added for count value. The Boundary for count: Min: 1, Max: 1000000 (10 Lakhs)
         2. For **Size** Option, an input field should be added for size value in kB(s). The Boundary for size: Min: 1 KB, Max: 20480 KB (20 MB).

### Save Type

1. **Advanced Options**: Optional for user, this contains some optional fields user can choose if required.
   1. **Delay:** A button that toggle delay field. If delay is enable user can set the delay the response. Boundaries for delay: Min: 0 seconds, Max: 60 seconds.

### Existing Type

* + 1. **Advanced Options**: Optional for user, this contains some optional fields user can choose if required.
       1. **Delay:** A button that toggle delay field. If delay is enable user can set the delay the response. Boundaries for delay: Min: 0 seconds, Max: 60 seconds.

## Use cases

* **Default**: Expected a RESTful data from within response. It may or may not be having dynamic response.
* **Save**: Expected a record to be create/update/edit/delete in to database
* **Existing**: Expected a fetched list of records stored during creation of save type.

## Requirements

1. User is able to create multiple mocks and able to use them in their existing project.
2. User is able to doing CRUD operation with the help of Mock APIs.
3. For any created mock service:
   1. If there is any custom headers added in the metadata of the API, the API will return those custom headers in API response.
   2. If there is any delay added to metadata of the API, the will response after n amount of seconds.
   3. If Combination of Mock Url and requested method does not match with stored data, it will follow the API Response standards and return 405, Method not allowed.
4. Mock Listing Page should contains sorting and pagination with limits of 10 records per page.
5. User Session must be persisted and cannot be clear on page refresh.