Developer Documnetation

Table of Content- Developer Documnetation

- Developer Documnetation
- Mock Api(s)
  - Mock List
    - Request
    - Query Params
  - Mock Create
  - Default Mock Create API
    - Request
    - Payload Options
    - Api Considerations
  - Save Mock Create API
    - Request
    - Payload Options
    - API Considrations
  - Existing Type Api
    - Payload Options
    - API Considrations
    - Response
  - Mock Detail
    - Request
    - Response
  - Mock Update
    - Request
    - Mock Status
    - Request
    - Payload Details
    - Response
- Project Api(s)
  - Project List
    - Request
    - Query Params
    - Response
  - Project Create
    - Request
    - Payload
    - Response
- JSON Upload
  - Response
- iRest (Simulator Api)
  - Request
  - Response

- Native Code to Generate the Database Collection Schema(s)

Mock Api(s)

Mock List

Request

```
var axios = require('axios');
var params = {
  pageNo,
  projectId,
  existing,
  userId,
};
var config = {
  method: 'get',
  url: 'http://localhost:9000/api/mock/list',
  headers: { }
};

axios(config)
.then(function (response) {
  console.log(JSON.stringify(response.data));
})
.catch(function (error) {
  console.log(error);
});
```

Query Params

Params	**Default Value**	**Description**
pageNo	1	Pagination Value
projectId	undefined	Optional, ProjectId belongs to the mock
existing	undefined	true, returns all mocks consists of serviceResponseType `save` POST records only
userId	undefined	Optional, return all mocks created by userId

Mock Create

There are 3 types of records allowed to be created:

- `Default`: Expected a REST data from the data in response
- `Save`: Expected a record to be stored/update/edit in to database
- `Existing`: Expecteded a fetched list of records stored during creation of `save` type.

Default Mock Create API

Request

```
var axios = require('axios');

var payload = {
    userId,
    projectId,
    serviceResponseType,
    serviceResponseBody,
    projectName,
    method,
    path,
    endpoint,
    statusCode,
    isDelay,
    delaySeconds,
    contentType,
    mockName,
    mockStatus,
    headers,
    params,
    isDynamicResponse,
    dynamicResponseKey,
    dynamicResponseRandom,
    dynamicResponseSpecific,
    dynamicResponseSpecificKeyValue,
    isDynamicImportCount,
    isDynamicImportSize,
    dynamicImportCount
};
var data = JSON.stringify(payload);

var config = {
  method: 'post',
  url: 'http://localhost:9000/api/mock/create',
  headers: {
    'Content-Type': 'application/json'
  },
  data : data
};

axios(config)
.then(function (response) {
  console.log(JSON.stringify(response.data));
})
.catch(function (error) {
  console.log(error);
});
```

Payload Options

Params	Description
userId	Mandatory, Id of the User
projectId	Mandatory, Id of the Project under which mock needs to be created
projectName	Mandatory, Name of Project under which mock is to be created
serviceResponseType	`default`
serviceResponseBody	Mandatory, A JSON Object expected from the response body
method	`GET` or `POST`
path	Mandatory, window.location.origin + MOCK_BASE_URL of the simulation API. MOCK_BASE_URL can be configured from .env
endpoint	Mandatory, API endpoint for eg. project-list or user-profile, etc.
statusCode	Mandatory, Expected Status Code from response. Available Values: `200`, `401`, `404`, `500`, `502`
isDelay	Mandatory, `true` or `false`
delaySeconds	Needed to specify with `isDelay: true`. An Integer Value between 0 to 60.
contentType	Mandatory, Content type of Api response. Available option: `Text/Plain`, `Application/Json`
mockName	Mandatory, Name of Mock
mockStatus	Mandatory, `enabled`/`disabled`
headers	An Array Consists of Headers For Eg. `[{'custom-header-1': 'custom-header-1-value'}, {'custom-header-2': 'custom-header-2-value'}]`. Default to []
params	An Array Consists of Params For Eg. `[{'searchQuery': 'test'}, {'page': 3}]`. Default to []
isDynamicResponse	Mandatory, `true` or `false`
dynamicResponseKey	Key name for which dynamic operations applied
dynamicResponseRandom	Set to `true` if dynamic value could be of any random type
dynamicResponseSpecific	Set to `true` if dynamic value could be of any specific type

Params	Description
dynamicResponseSpecificKeyValue	Could be an array, string or object value that needed to pushed to the `dynamicResponseKey`value with dynamic operations
isDynamicImportCount	Set to `true` if number of dynamic records needed to set
dynamicImportCount	Count of `dynamicResponseKey` value used with `isDynamicImportCount`. For eg. 100 or 245. Maximum is 1000000 records
isDynamicImportSize	Set to `true` if size of dynamic records needed to set
dynamicImportSize	Size of `dynamicResponseKey` value used with `isDynamicImportSize`. For eg. 50 or 500 in KB. Maximum is 20971520 KB (20 MB)

Api Considerations

- We can only enable one kind of dynamic data value in single Api. In another words, dynamic response value could be random or specific i.e. either `dynamicResponseRandom` can be `true` or `dynamicResponseSpecific` can be `true`.
- We can only enable either dynamic count or dynamic size in single Api i.e. `isDynamicImportCount` can be `true` or `isDynamicImportSize` can be `true`.
- `dynamicImportCount` key is needed only if `isDynamicImportCount` is `true`. The size is limited to 1000000 (10 Lakhs) records per Api Call. Also, if `0 < dynamicImportCount <= 100000 (1 Lakh)`, the data is send as normal JSON response however if `100000 (1 Lakh) < dynamicImportCount <= 1000000 (10 Lakh)`, the data will be convert into a stream format with additional response headers `'Transfer-Encoding': 'chunked'`.
- `dynamicImportSize` key is needed only if `isDynamicImportSize` is `true`. The size is limited to 1000000 (10 Lakhs) records per Api Call. Also, if `0 < dynamicImportSize <= 1024 KB (1 MB)`, the data is send as normal JSON response however if `1024 KB (1 MB) < dynamicImportSize <= 20480 KB (20 MB)`, the data will be convert into a stream format with additional response headers `'Transfer-Encoding': 'chunked'`.
- `delaySeconds` key is needed only if `isDelay` is `true`
- A Record for `default` type Api is stored in serviceResponse database collection after metadata stored in mock database collection

Save Mock Create API

Request

```
var axios = require('axios');
var payload = {
    userId,
    projectId,
        serviceResponseType,
        projectName,
        method,
        path,
        endpoint,
        statusCode,
        isDelay,
        delaySeconds,
        contentType,
        mockName,
        mockStatus,
        headers,
        params,
        isDynamicResponse,
    };
    var data = JSON.stringify(payload);

    var config = {
      method: 'post',
      url: 'http://localhost:9000/api/mock/create',
      headers: {
        'Content-Type': 'application/json'
      },
      data : data
    };

    axios(config)
    .then(function (response) {
      console.log(JSON.stringify(response.data));
    })
    .catch(function (error) {
      console.log(error);
    });
```

Payload Options

Params	Description
userId	Mandatory, Id of the User
projectId	Mandatory, Id of the Project under which mock needs to be created
projectName	Mandatory, Name of Project under which mock is to be created
serviceResponseType	`save`
method	`POST`, `PUT`, `PATCH`, `DELETE`
path	Mandatory, window.location.origin + MOCK_BASE_URL of the simulation API. MOCK_BASE_URL can be configured from .env
endpoint	Mandatory, API endpoint for eg. project-list or user-profile, etc.

Params	Description
statusCode	Mandatory, Expected Status Code from response. Available Values: `200`, `401`, `404`, `500`, `502`
isDelay	Mandatory, `true` or `false`
delaySeconds	Needed to specify with `isDelay: true`. An Integer Value between 0 to 60.
contentType	Mandatory, Content type of Api response. Available option: `Text/Plain`, `Application/Json`
mockName	Mandatory, Name of Mock
mockStatus	Mandatory, `enabled`/`disabled`
headers	An Array Consists of Headers For Eg. `[{'custom-header-1': 'custom-header-1-value'}, {'custom-header-2': 'custom-header-2-value'}]`. Default to []
params	[]
isDynamicResponse	`false`

API Considrations

- POST Method
  - Stores a record in JSON/Plain-Text format in serviceResponse table
- PUT Method
  - Overwriting the JSON/Plain-Text record created in save POST Api.
- PATCH method
  - Update the JSON record created in save POST Api.
  - This updates JSON record only. This method does not work with Plain-Text.
- DELETE Method
  - Delete a JSON/Plain-Text record created in save POST Api.
- During creation the of new Api, apart from basic validation, we are checking the **combination of endpoint and method is unique** in database. This can be done with indexding the combination of two fields in mongodb. If the combination is not unique, the Api will throw error for duplicate Api. In other words, Api can be creating on same endpoint but with different methods.
- A Record for `save` type Api is stored in serviceResponse database collection during user try to hit POST Api and stores the data. Once, the data is stored user can overide, modify or remove the response with `PUT`, `PATCH`, `DELETE` method, respectively.
- In some scenerios, user may overwrite the Api metadata, if that is the required step, we are deleted all the stored data related to it from serviceResponse table.

Existing Type Api

```
var axios = require('axios');
var payload = {
    userId,
    projectId,
    serviceResponseType,
    referenceId,
    projectName,
    method,
    path,
    endpoint,
    statusCode,
    isDelay,
    delaySeconds,
    contentType,
    mockName,
    mockStatus,
    headers,
    params,
    isDynamicResponse,
};
var data = JSON.stringify(payload);

var config = {
  method: 'post',
  url: 'http://localhost:9000/api/mock/create',
  headers: {
    'Content-Type': 'application/json'
  },
  data : data
};

axios(config)
.then(function (response) {
  console.log(JSON.stringify(response.data));
})
.catch(function (error) {
  console.log(error);
});
```

Payload Options

Params	Description
userId	Mandatory, Id of the User
projectId	Mandatory, Id of the Project under which mock needs to be created
projectName	Mandatory, Name of Project under which mock is to be created
serviceResponseType	`existing`

Params	Description
referenceId	Mandatory, ReferenceId of POST Api of `save` type created by same user under same project
method	`GET`
path	Mandatory, window.location.origin + MOCK_BASE_URL of the simulation API. MOCK_BASE_URL can be configured from .env
endpoint	Mandatory, API endpoint for eg. project-list or user-profile, etc.
statusCode	Mandatory, Expected Status Code from response. Available Values: `200`, `401`, `404`, `500`, `502`
isDelay	Mandatory, `true` or `false`
delaySeconds	Needed to specify with `isDelay: true`. An Integer Value between 0 to 60.
contentType	`Application/Json`
mockName	Mandatory, Name of Mock
mockStatus	Mandatory, `enabled`/`disabled`
headers	An Array Consists of Headers For Eg. `[{'custom-header-1': 'custom-header-1-value'}, {'custom-header-2': 'custom-header-2-value'}]`. Default to []
params	[]
isDynamicResponse	`false`

API Considrations

- `ReferenceId` is pointed to another mock Id. Mock on which it is refer, had been of `save` type. It should be created by same user and will be under same project.
- The Api return the list of existing records. All the existing records has been stored from POST Api usage, created under `save` type of Api.

Response

```
// success
{
    "message": "Mock created successfully.",
    "status": 201,
    "data": {
        id,
        mockName,
        mockUrl,
    }
}
```

```
{ // duplicate endpoint
    "message": "Endpoint already exists. Choose different endpoint.",
    "status": 417
}
```

```
// If user not exist in records
"message": "User does not exists.",
"status": 400
```

```
// If project not exist in records
"message": "Project does not exists.",
"status": 400
```

```
// If dynamicResponseSpecificKeyValue is invalid object or array
"message": "Invalid key dynamicResponseSpecificKeyValue.",
"status": 400
```

```
// If serviceResponseBody is invalid object
"message": "Invalid key serviceResponseBody.",
"status": 400
```

Mock Detail

Request

```
var axios = require('axios');

var config = {
  method: 'get',
  url: 'http://localhost:9000/api/mock/item/:mockId',
  headers: { }
};

axios(config)
```

```
.then(function (response) {
  console.log(JSON.stringify(response.data));
})
.catch(function (error) {
  console.log(error);
});
```

Response

Scenerio 1: `default` *serviceResponseType*

```
{
  "message": "Mock Detail",
  "status": 200,
  "data": {
    mock: {
      status,
      serviceResponseType,
      path,
      method,
      statusCode,
      isDelay,
      contentEncoding
      contentType,
      mockStatus
      isDynamicResponse,
      _id
      projectId,
      projectName,
      userId,
      endpoint,
      delaySeconds,
      headers: [
        {
          key,
          value
        }
      ],
      params: [
        {
          key,
          value,
        }
      ],
      mockName,
      dynamicResponseKey,
      dynamicResponseSpecific,
      dynamicResponseSpecificKeyValue,
      isDynamicImportSize,
      dynamicImportSize,
      endpointRequestPath,
      createdAt,
      updatedAt,
      serviceResponse: {
          status,
          _id,
          serviceResponseBody,
          mockId,
          contentType,
          createdAt,
          updatedAt,
      }
    }
  }
}
```

Scenerio 2: save serviceResponseType

```
{
  "message": "Mock Detail",
  "status": 200,
  "data": {
    mock: {
      status,
      serviceResponseType,
      path,
      method,
      statusCode,
      isDelay,
      contentEncoding
      contentType,
      mockStatus
      isDynamicResponse,
      _id
      projectId,
      projectName,
      userId,
      endpoint,
      delaySeconds,
      headers: [
        {
          key,
          value
        }
      ],
      params: [
        {
          key,
          value,
        }
      ],
      mockName,
      endpointRequestPath,
      createdAt,
      updatedAt,
    }
  }
}
```

Scenerio 3: `existing` *serviceResponseType*

```
  {
    "message": "Mock Detail",
    "status": 200,
    "data": {
      mock: {
        status,
        serviceResponseType,
        path,
        method,
        statusCode,
        isDelay,
        contentEncoding
        contentType,
        mockStatus
        isDynamicResponse,
        _id
        projectId,
        referenceId,
        projectName,
        userId,
        endpoint,
        delaySeconds,
        headers: [
          {
            key,
            value
          }
        ],
        params: [
          {
            key,
            value,
          }
        ],
        mockName,
        endpointRequestPath,
        createdAt,
        updatedAt,
      }
    }
  }
```

Mock Update

Request

Request Url for Mock Update: `http://localhost:9000/api/mock/item/:mockId`.

- *Important Note*: During updation of any mock, all the serviceResponse(s) stored related to will be delete, irrespective of the serviceResponseType. It means, user will not be able to view older data once they update the Api metadata.
- User should not be able to update mock endpoint during the updation however it is available in request payload for other references.
- For Other *Request*, *Payload*, *Api consideration* and *Response*, refer the Mock Create Api Section

Mock Status

Request

```
var axios = require('axios');
var payload = {
  mockStatus,
}
var data = JSON.stringify(payload);

var config = {
  method: 'patch',
  url: 'http://localhost:9000/api/mock/item/:mockId',
  headers: {
    'Content-Type': 'application/json'
  },
  data : data
};

axios(config)
.then(function (response) {
  console.log(JSON.stringify(response.data));
})
.catch(function (error) {
  console.log(error);
});
```

Payload Details

Params	Description

Params	Description
mockStatus	`enabled` or `disabled`

Response

```
// success
{
  "message": "Status changes succesfully.",
  "status": 200,
  data: {
    mock: {
      status,
      // remaining metadata
    }
  }
}
```

Project Api(s)

Project List

Request

```
var axios = require('axios');
var params = {
  PageNo,
  searchQuery,
  userId,
};
var config = {
  method: 'get',
  url: 'http://localhost:9000/api/project/list',
  headers: { }
  params,
};

axios(config)
.then(function (response) {
  console.log(JSON.stringify(response.data));
})
.catch(function (error) {
  console.log(error);
});
```

Query Params

Params	Default Value	Description
PageNo	1	Pagination Value
searchQuery	undefined	Optional
userId	undefined	optional

Response

```
"status": 200, // success
"message": "Project List",
"data": {
  "projectList": [
    {
        _id,
        projectName,
    },
  ]
}
```

- API will return 204 No Data Found in case no project match the criteria

Project Create

Request

```
var axios = require('axios');
var data =
JSON.stringify({"userId":"5ee9bb30ae65d45d0831011f","projectName":"irest-
trial2"});

var config = {
  method: 'post',
  url: 'http://localhost:9000/api/project/create',
  headers: {
    'Content-Type': 'application/json'
  },
  data : data
};

axios(config)
.then(function (response) {
  console.log(JSON.stringify(response.data));
})
.catch(function (error) {
  console.log(error);
});
```

Payload

Params	Description
userId	Mandatory
projectName	Name of project

Response

```
  "status": 201, // success
  "message": "Project created Successfully.",

  "data": {
    _id // projectId of new create project
    projectList" // all project List created by user
  }
```

```
  // If user not exist in records
  "message": "User does not exists.",
  "status": 400
```

```
  status: 417 // duplicate
  message: "Project Already Exist.",
```

JSON Upload

Request

```
  var axios = require('axios');
  var FormData = require('form-data');
  var fs = require('fs');
```

```
var data = new FormData();
data.append('file', fs.createReadStream('large_data_file.json'));

var config = {
  method: 'post',
  url: 'http://localhost:9000/api//file/json-upload',
  headers: {
    ...data.getHeaders()
  },
  data : data
};

axios(config)
.then(function (response) {
  console.log(JSON.stringify(response.data));
})
.catch(function (error) {
  console.log(error);
});
```

Response

```
// success
{
  "message": "JSON FILE DATA",
  "status": 200,
  data // JSON Data body
}
```

```
{
  "message": "File type must be of JSON type.", // if non-json file is being
uploaded
  "status": 400
}
```

```
{
  "message": "Unable to Parse JSON.", // Invalid JSON Format Data
  "status": 400
}
```

iRest (Simulator Api)

Request

```
var axios = require('axios');

var config = {
  method: <any>,
  url: 'http://localhost:9000/api/rest/:projectName/:endpoint',
  headers: { }
};

axios(config)
.then(function (response) {
  console.log(JSON.stringify(response.data));
})
.catch(function (error) {
  console.log(error);
});
```

Response

There are multiple usecases for end user, according to the different usecase different kind of response user will get.

There are some Common usecases which are applied to app simulation Api.

Common Usecase 1

No Mock Data

Returns a 404 response with no Matching Mock Found.

Common Usecase 2

Non Matching Mock

If Mock Url matched with the requested Url but requested method does not match, it will follow the Api Response standards and return 405, Method not allowed.

Common Usecase 3

Disabled Mock

If Mock status is found in disabled state, User will get 404 error with message mock is disabled.

Common Usecase 4

Custom Headers

If there is any custom headers added in the metadata of the Api, the Api will return those custom headers in Api response.

Common Usecase 5

Delay

If there is any delay added to metadata of the Api, the will response after n amount of seconds. Maximum delay for the Api Response is 60 Seconds (1 Minute).

Common Usecase 6

If the requested methods is not found in the allowed method list, the Api will simply returns 404 Not Found. This usecase is similar to Common Usecase 2 mentioned above however the behaviour is different.

There are 3 types of records for which iRest simulate the response:

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

Default Response Type

Config

Parameter	**Description**
Method	`GET` or `POST`
projectName	Name of project under which your mock is created
endpoint	endpoint of mock, stored in metadata when mock is created from create Api.

Default Response Type Usecase 1

Normal Response: Returns the JSON/Plain-Text stored in service response collection for corresponding record.

Default Response Type Usecase 2 (applicable for Content-Type: Application/json)

- Dynamic Response For Specific Key (Random Key Value & Dynamic Count): Returns the response with dynamic records value of JSON Key mentioned.
    - The Value of Key has multiple records as mentioned in count value.
    - The Value type is generally long string values containing n fake entries, where n is the count value.
    - The data count below 100000 (1 Lakh) records sends as normal json response.
    - The data count above 100000 (1 Lakh) records upto 1000000 (10 Lakhs) records will be sended as data stream with additional headers `'Transfer-Encoding': 'chunked'`
    - We are not entertaining data count above 1000000 (10 Lakh).
- Dynamic Response For Specific Key (Random Key Value & Dynamic Size): Returns the response with dynamic records value of JSON Key mentioned.
    - The Value of Key has multiple records of mentioned KBs of data.
    - The Value type is generally long string values containing fake entries.
    - The data size below 1024 KB (1 MB) records sends as normal json response.
    - The data size above 100000 (1 MB) records upto 20480 KB (20 MB) records will be sended as data stream with additional headers `'Transfer-Encoding': 'chunked'`
    - We are not entertaining data size above 20480 KB (20 MB).
- Dynamic Response For Specific Key (Specific Key Value & Dynamic Count): Returns the response with dynamic records value of JSON Key mentioned.
    - The Value of Key has multiple records as mentioned in count value.
    - The Value type is mentioned Json/Object/String type values mentioned in keyValue containing n entries, where n is the count value.
    - The data count below 100000 (1 Lakh) records sends as normal json response.
    - The data count above 100000 (1 Lakh) records upto 1000000 (10 Lakhs) records will be sended as data stream with additional headers `'Transfer-Encoding': 'chunked'`
    - We are not entertaining data count above 1000000 (10 Lakh).
- Dynamic Response For Specific Key (Specific Key Value & Dynamic Size): Returns the response with dynamic records value of JSON Key mentioned.
    - The Value of Key has multiple records of mentioned KBs of data.
    - The Value type is mentioned Json/Object/String type values mentioned in keyValue entries
    - The data size below 1024 KB (1 MB) records sends as normal json response.
    - The data size above 100000 (1 MB) records upto 20480 KB (20 MB) records will be sended as data stream with additional headers `'Transfer-Encoding': 'chunked'`
    - We are not entertaining data size above 20480 KB (20 MB).

Dynamic Key Values Options

- Dynamic Key of array type & dynamic key value of object type
  - Dynamic Key: `answers: []`
  - Dynamic Value: `{'answer': 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'}`

Generated Dynmaic Value

```
  answers: [
    {'answer': 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'},
    {'answer': 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'},
    {'answer': 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'},
    {'answer': 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'},
    {'answer': 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'
    }
  ]
```

- Dynamic Key of array type & dynamic key value of array type
  - Dynamic Key: `answers: []`
  - Dynamic Value: `[{'answer1': 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'}, {'answer'2: 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'}]`

Generated Dynmaic Value

```
  answers: [
    [{'answer1': 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'}, {'answer'2: 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'}],
    [{'answer1': 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'}, {'answer'2: 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'}],
    [{'answer1': 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'}, {'answer'2: 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'}],
    [{'answer1': 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'}, {'answer'2: 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'}],
    [{'answer1': 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'}, {'answer'2: 'Lorem
```

```
ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua.'}]
  ]
```

- Dynamic Key of array type & dynamic key value of string type
  - Dynamic Key: `answers: {}`
  - Dynamic Value: `'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'`

Generated Dynmaic Value

```
  answers: [
    'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua.',
    'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua.',
    'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua.',
    'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua.',
    'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua.'
  ]
```

- Dynamic Key of object type & dynamic key value of object type
  - Dynamic Key: `answers: {}`
  - Dynamic Value: `{'answer': 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'}`

Generated Dynmaic Value

```
  answers: {
    0: {'answer': 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua.'},
    1: {'answer': 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua.'},
    2: {'answer': 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua.'},
    3: {'answer': 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua.'},
    4: {'answer': 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua.'}
  }
```

- Dynamic Key of object type & dynamic key value of array type
  - Dynamic Key: `answers: {}`

- Dynamic Value: `[{'answer1': 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'}, {'answer'2: 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'}]`

Generated Dynmaic Value

```
  answers: {
    0: [{'answer1': 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'}, {'answer'2: 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'}],
    1: [{'answer1': 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'}, {'answer'2: 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'}],
    2: [{'answer1': 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'}, {'answer'2: 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'}],
    3: [{'answer1': 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'}, {'answer'2: 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'}],
    4: [{'answer1': 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'}, {'answer'2: 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'}]
  }
```

- Dynamic Key of object type & dynamic key value of string type
  - Dynamic Key: `answers: []`
  - Dynamic Value: `'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'`

Generated Dynmaic Value

```
  answers: {
    0: 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.',
    1: 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.',
    2: 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.',
    3: 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.',
    4: 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'
  }
```

- Dynamic Key of string type & dynamic key value of object type
    - Dynamic Key: `answers: ''`
    - Dynamic Value: `{'answer': 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'}`

Generated Dynmaic Value

```
  answers: '
    {answer: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua.}
    {answer: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua.}
    {answer: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua.}
    {answer: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua.}
    {answer: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua.}
  '
```

- Dynamic Key of string type & dynamic key value of array type
    - Dynamic Key: `answers: ''`
    - Dynamic Value: `[{'answer1': 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'}, {'answer'2: 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'}]`

Generated Dynmaic Value

```
  answers: '
    [{answer: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua.}]
    [{answer: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua.}]
    [{answer: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua.}]
    [{answer: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua.}]
    [{answer: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua.}]
  '
```

- Dynamic Key of string type & dynamic key value of string type
    - Dynamic Key: `answers: ''`
    - Dynamic Value: `'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'`

Generated Dynmaic Value

```
  answers: '
    Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua.
    Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua.
    Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua.
    Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua.
    Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua.
  '
```

Save Response Type

Config

Parameter	Description
Method	`POST`, `PUT`, `PATCH`, `DELETE`
projectName	Name of project under which your mock is created
endpoint	endpoint of mock, stored in metadata when mock is created from create Api.

There are 4 types of Api user can create and use in their project

- POST Method
  - Stores a record in JSON/Plain-Text format in serviceResponse table
- PUT Method
  - Overwriting the JSON/Plain-Text record created in save POST Api.
- PATCH method
  - Update the JSON record created in save POST Api.
  - This updates JSON record only. This method does not work with Plain-Text.
- DELETE Method
  - Delete a JSON/Plain-Text record created in save POST Api.

Existing Response Type

Config

Parameter	Description
Method	`GET`
projectName	Name of project under which your mock is created

Parameter	Description
endpoint	endpoint of mock, stored in metadata when mock is created from create Api.

There is only single usecase for existing Api.

- Existing Api will returns the list of records end-user stored for `save` type Api.

Native Code to Generate the Database Collection Schema(s)

Properties

```
// Project Collection
db.createCollection('projects', {
  validator: {
    $jsonSchema: {
      bsonType: 'object',
      title: 'projects',
      required: ['status', 'projectName', 'userId', 'createdAt', 'updatedAt',
'__v'],
      properties: {
        status: {
          bsonType: 'string'
        },
        projectName: {
          bsonType: 'string'
        },
        userId: {
          bsonType: 'objectId'
        },
        createdAt: {
          bsonType: 'date'
        },
        updatedAt: {
          bsonType: 'date'
        },
        __v: {
          bsonType: 'double'
        }
      }
    }
  }
});

// User Collection
db.createCollection('users', {
  validator: {
    $jsonSchema: {
      bsonType: 'object',
      title: 'users',
      required: ['status', 'name', 'email', 'password', 'createdAt', 'updatedAt',
'__v'],
      properties: {
        status: {
          bsonType: 'string'
        },
        name: {
          bsonType: 'string'
        },
        email: {
          bsonType: 'string'
        },
        password: {
          bsonType: 'string'
        },
        createdAt: {
          bsonType: 'date'
        },
        updatedAt: {
          bsonType: 'date'
        },
        __v: {
          bsonType: 'double'
        }
      }
    }
  }
});

// Service Response
db.createCollection('serviceresponses', {
  validator: {
    $jsonSchema: {
      bsonType: 'object',
      title: 'serviceresponses',
      required: ['status', 'serviceResponseBody', 'mockId', 'createdAt',
'updatedAt', '__v', 'contentType'],
      properties: {
        status: {
          bsonType: 'string'
        },
        serviceResponseBody: {
          bsonType: 'string'
        },
        mockId: {
          bsonType: 'objectId'
        },
        createdAt: {
          bsonType: 'date'
        },
        updatedAt: {
          bsonType: 'date'
        },
        __v: {
          bsonType: 'double'
        },
        contentType: {
          bsonType: 'string'
        }
      }
    }
  }
});

// Mocks Collection
db.createCollection('mocks', {
  validator: {
    $jsonSchema: {
      bsonType: 'object',
      title: 'mocks',
      required: ['status', 'serviceResponseType', 'path', 'method', 'statusCode',
'isDelay', 'contentEncoding', 'contentType', 'mockStatus', 'isDynamicResponse',
'userId', 'projectId', 'projectName', 'endpoint', 'delaySeconds', 'mockName',
'headers', 'params', 'dynamicResponseKey', 'dynamicResponseSpecific',
'dynamicResponseSpecificKeyValue', 'isDynamicImportSize', 'dynamicImportSize',
'endpointRequestPath', 'createdAt', 'updatedAt', '__v', 'isDynamicImportCount',
'dynamicImportCount', 'dynamicResponseRandom', 'referenceId'],
      properties: {
        status: {
          bsonType: 'string'
        },
        serviceResponseType: {
          bsonType: 'string'
        },
        path: {
          bsonType: 'string'
        },
        method: {
          bsonType: 'string'
        },
        statusCode: {
          bsonType: 'string'
        },
        isDelay: {
          bsonType: 'bool'
        },
        contentEncoding: {
          bsonType: 'string'
        },
        contentType: {
          bsonType: 'string'
        },
        mockStatus: {
          bsonType: 'string'
        },
        isDynamicResponse: {
          bsonType: 'bool'
        },
        userId: {
          bsonType: 'objectId'
        },
        projectId: {
          bsonType: 'objectId'
        },
        projectName: {
          bsonType: 'string'
        },
        endpoint: {
          bsonType: 'string'
        },
        delaySeconds: {
          bsonType: 'double'
        },
        mockName: {
          bsonType: 'string'
        },
        headers: {
          bsonType: 'array',
          items: {
            title: 'mocks.headers',
            required: ['key', 'value'],
            properties: {
              key: {
                bsonType: 'string'
              },
              value: {
                bsonType: 'string'
              }
            }
          }
        },
        params: {
          bsonType: 'array',
          items: {
            title: 'mocks.params',
            required: ['key', 'value'],
            properties: {
              key: {
                bsonType: 'string'
              },
              value: {
                bsonType: 'string'
              }
            }
          }
        },
        dynamicResponseKey: {
          bsonType: 'string'
        },
        dynamicResponseSpecific: {
          bsonType: 'bool'
        },
        dynamicResponseSpecificKeyValue: {
          bsonType: 'string'
        },
        isDynamicImportSize: {
          bsonType: 'bool'
        },
        dynamicImportSize: {
          bsonType: 'double'
        },
        endpointRequestPath: {
          bsonType: 'string'
        },
        createdAt: {
          bsonType: 'date'
        },
        updatedAt: {
          bsonType: 'date'
        },
        __v: {
          bsonType: 'double'
        },
        isDynamicImportCount: {
          bsonType: 'bool'
        },
        dynamicImportCount: {
          bsonType: 'double'
        },
        dynamicResponseRandom: {
          bsonType: 'bool'
        },
        referenceId: {
          bsonType: 'objectId'
        }
      }
    }
  }
});
```