**BACKEND SERVICE TECH STACKS:**

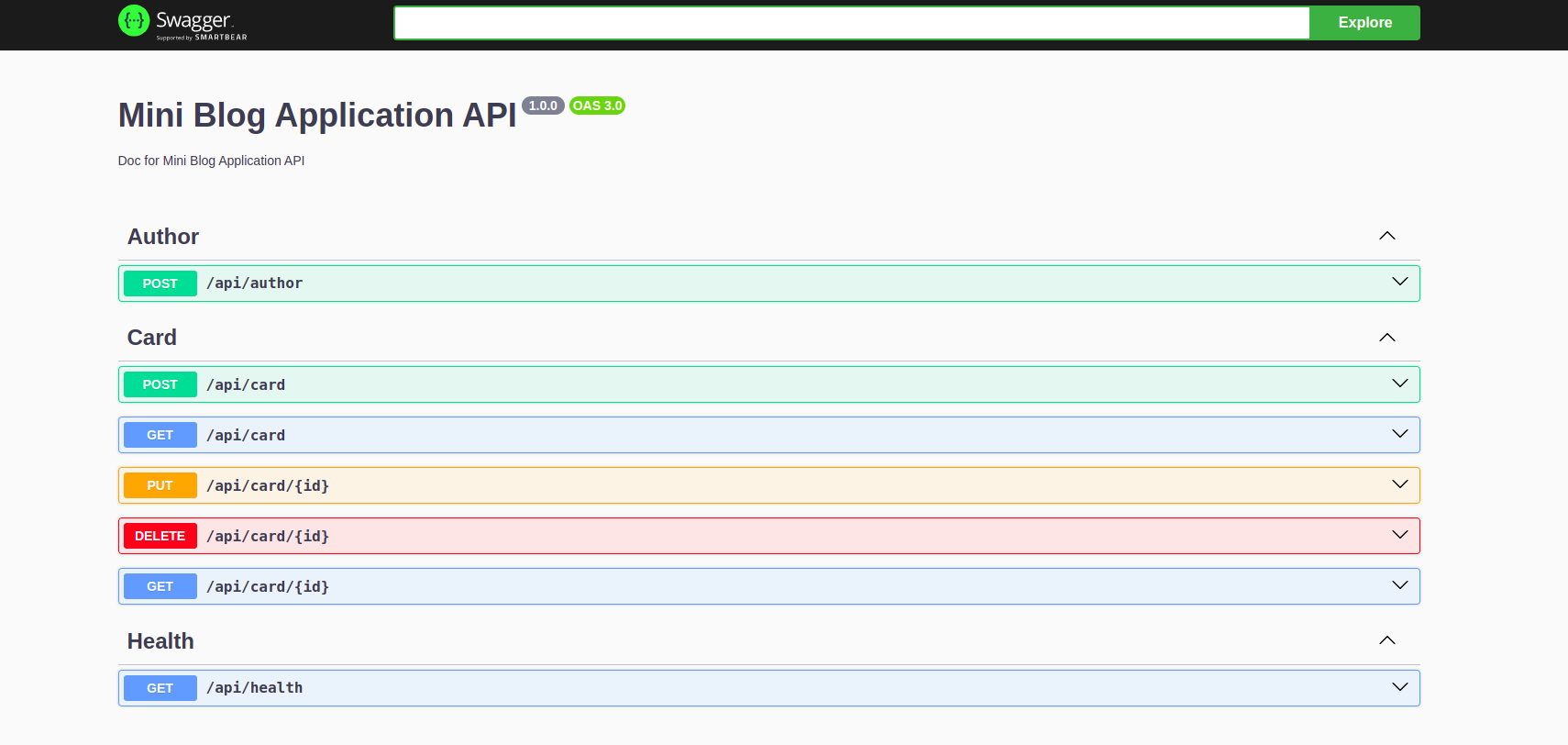
| **S.N.** | **SERVICE** | **STACK** |
| --- | --- | --- |
| 1. | BACKEND | NODEJS KOA Framework |
| 2. | DATABASE | MONGO DB |

**FIGURES: Can be referenced in /figuration folder of** [**mini-blog**](https://github.com/aishwaryawambule/mini-blog) **application**

**GIT REPO LINK:** [**https://github.com/aishwaryawambule/mini-blog**](https://github.com/aishwaryawambule/mini-blog)

**API SWAGGER LINK:** [**http://localhost:5005/swagger**](http://localhost:5005/swagger)

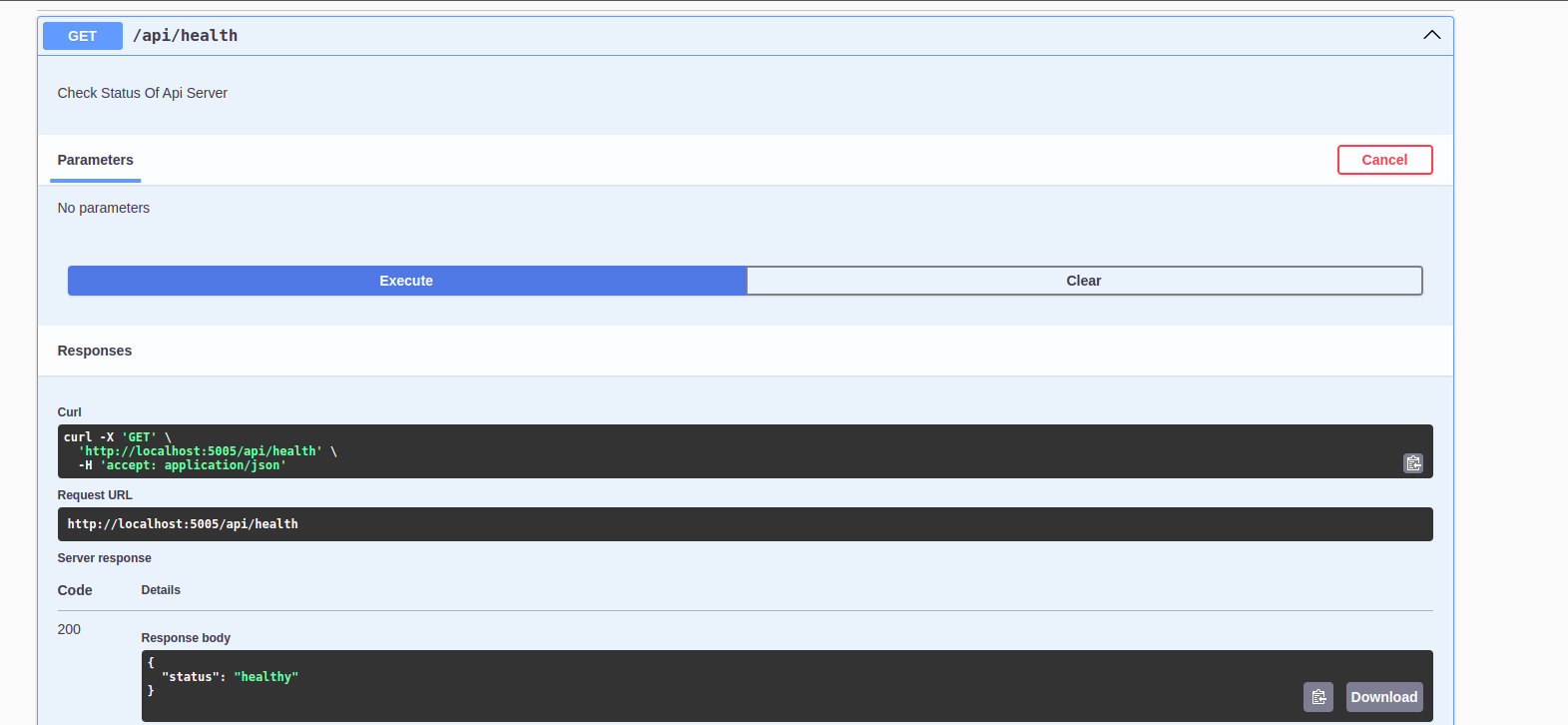
**API SWAGGER IMAGE DEMO:**



**API Function Description:**

1. **GET: /api/health:**

Provides health status of the server.



1. **POST: /api/Author:**

Initial API to create author.

Request Body:

{

userName,

emailAddress

}

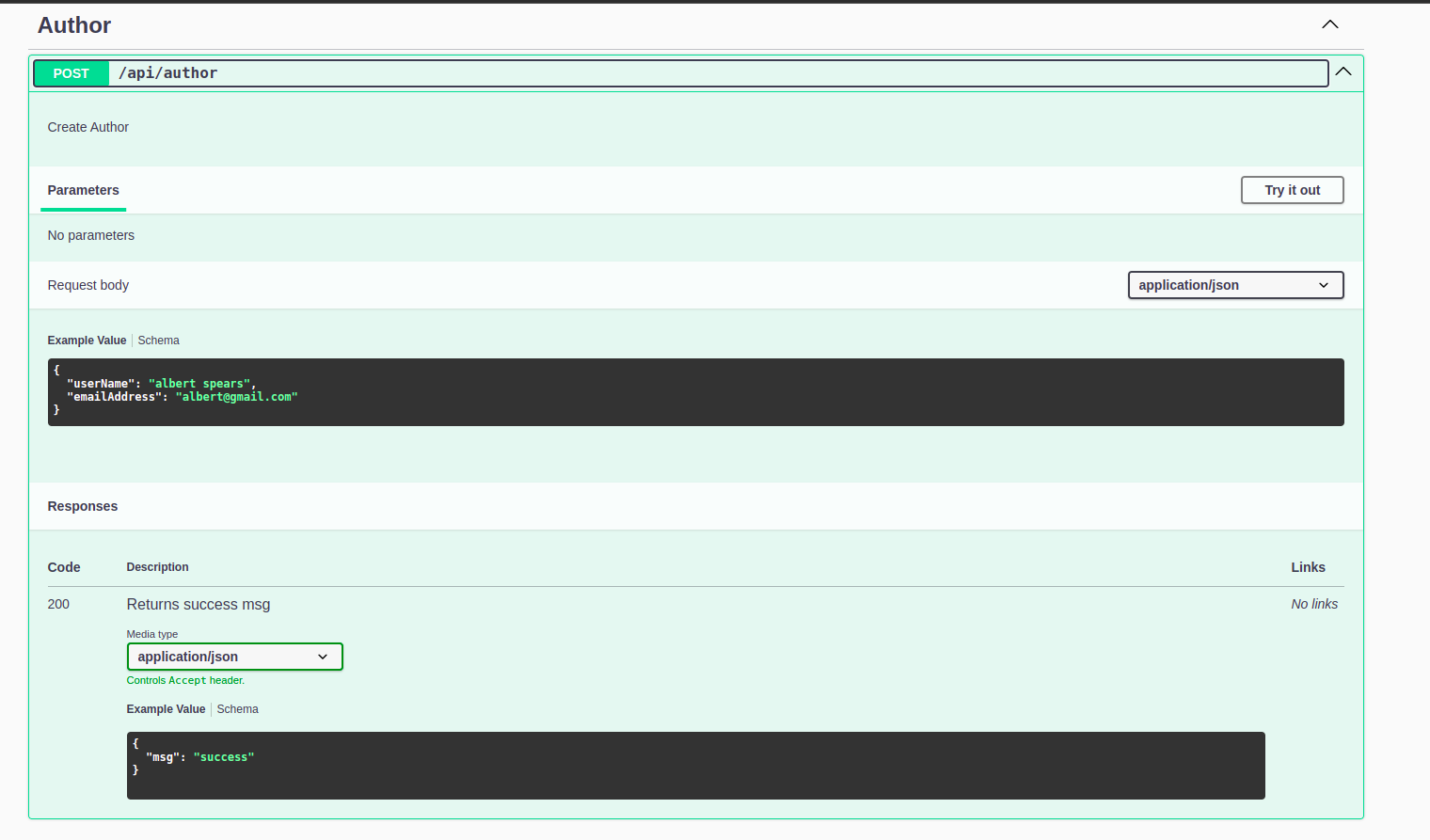
API method handles:

* Request Body validation
* Email address validation
* Duplicate email validation

Response:

- Success Message for success case

- Error Message for failure case



1. **POST: /api/card:**

API to create a card.

//the status of the card is set as NEW when the card is created.

Request Body:

{

name, //card name

emailAddress, //author email address

Category, // card category

Description // content description

}

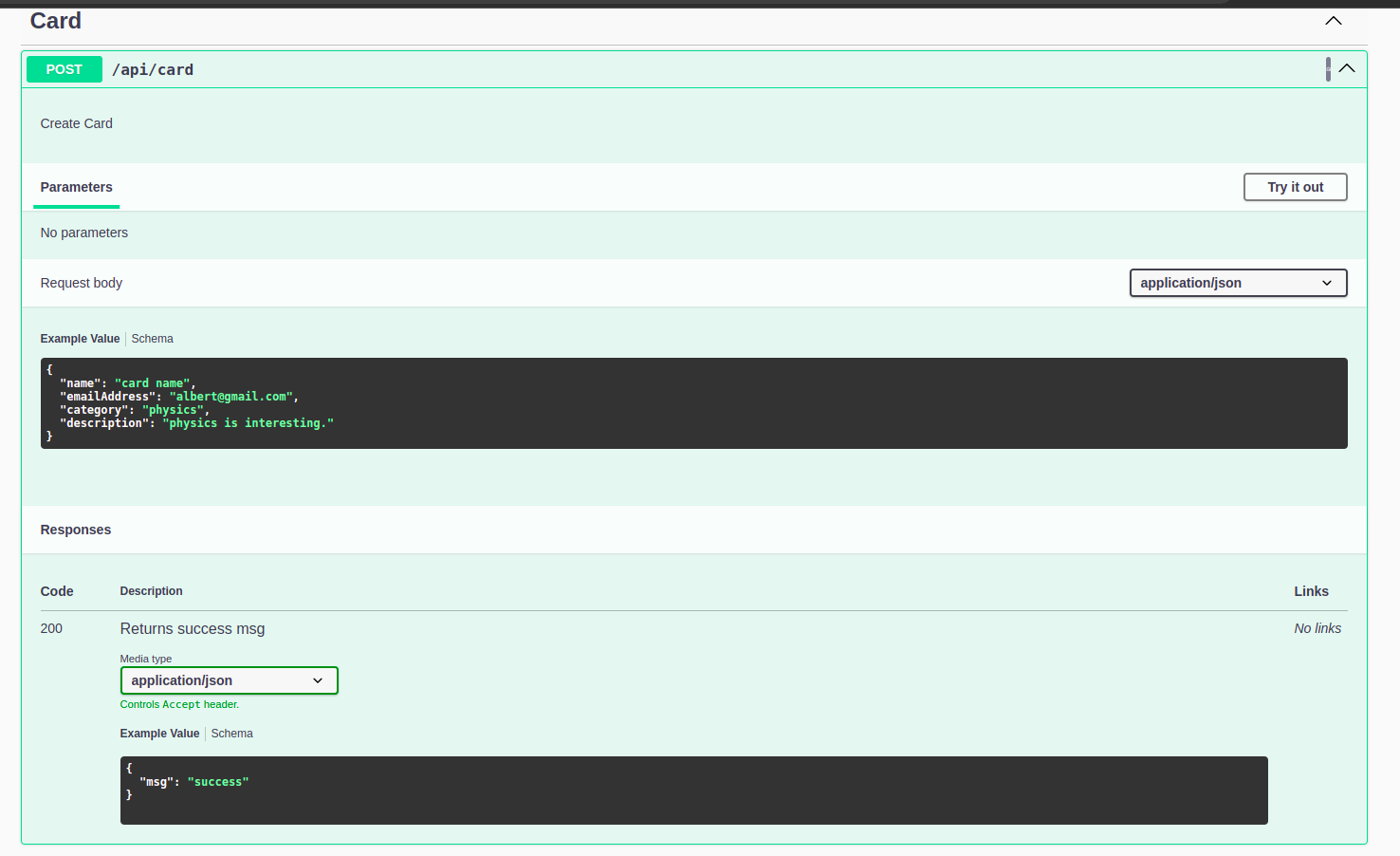
API method handles:

* Request Body validation
* Email address validation
* Check whether the Author Exists by the provided email address
* Create Card

Response:

- Success Message for success case

- Error Message for failure case



1. **GET: /api/card**

Fetch all card details by pagination limitation.

Request Parameters:

* limit
* page

API method handles:

* Fetch Card, Content, Author Details

Response:

- Response Body for Success Case:

[ {

id,

name,

status,

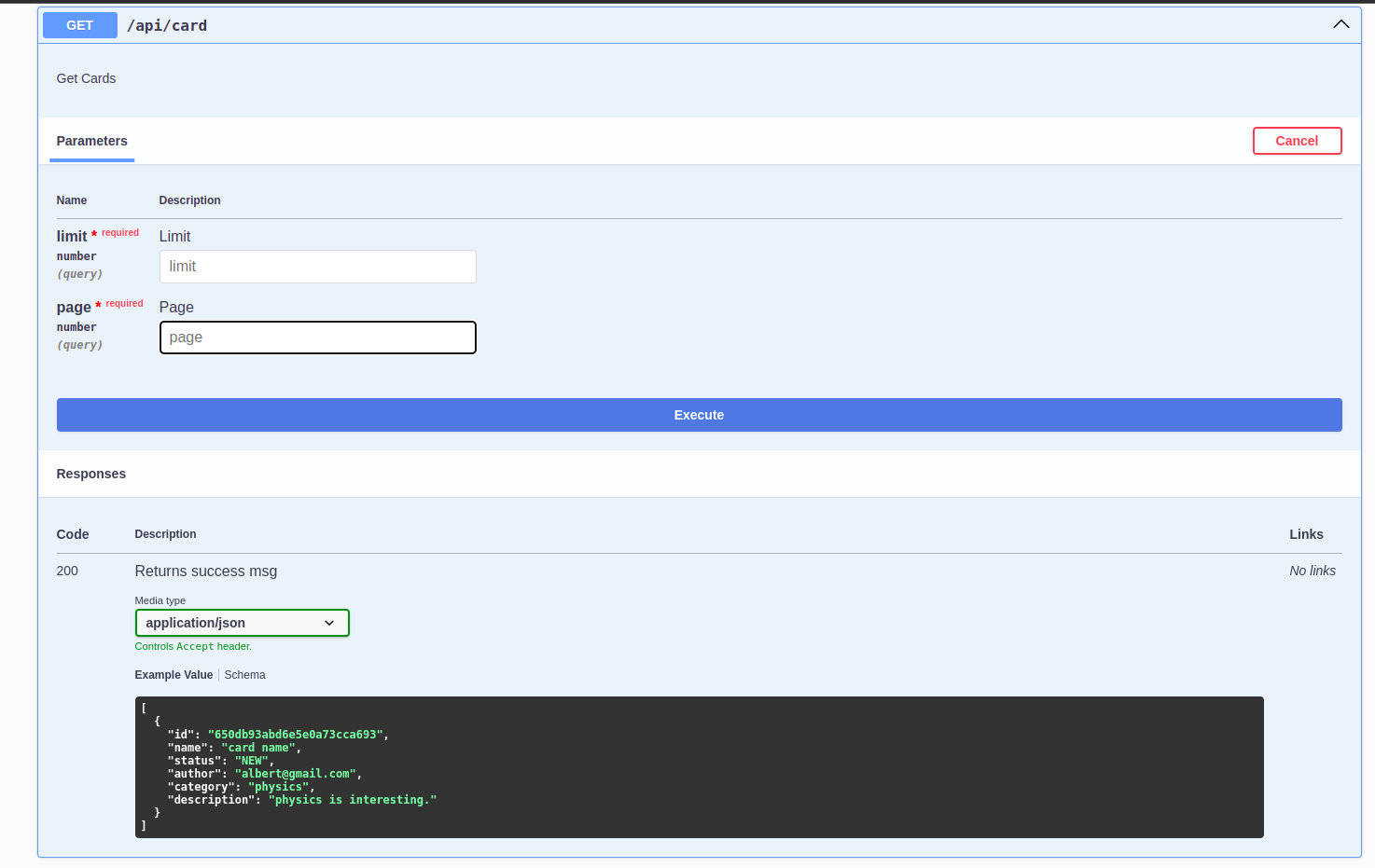
author,

category,

description

} ]

- Error Message for failure case

****

1. **PUT: /api/card/{id}**

Update card details by Card Id.

//the status of the card is set as EDITED when the card is edited.

Request Parameter:

* id // cardId

Request Body:

{

name,

emailAddress,

category,

description

}

API method handles:

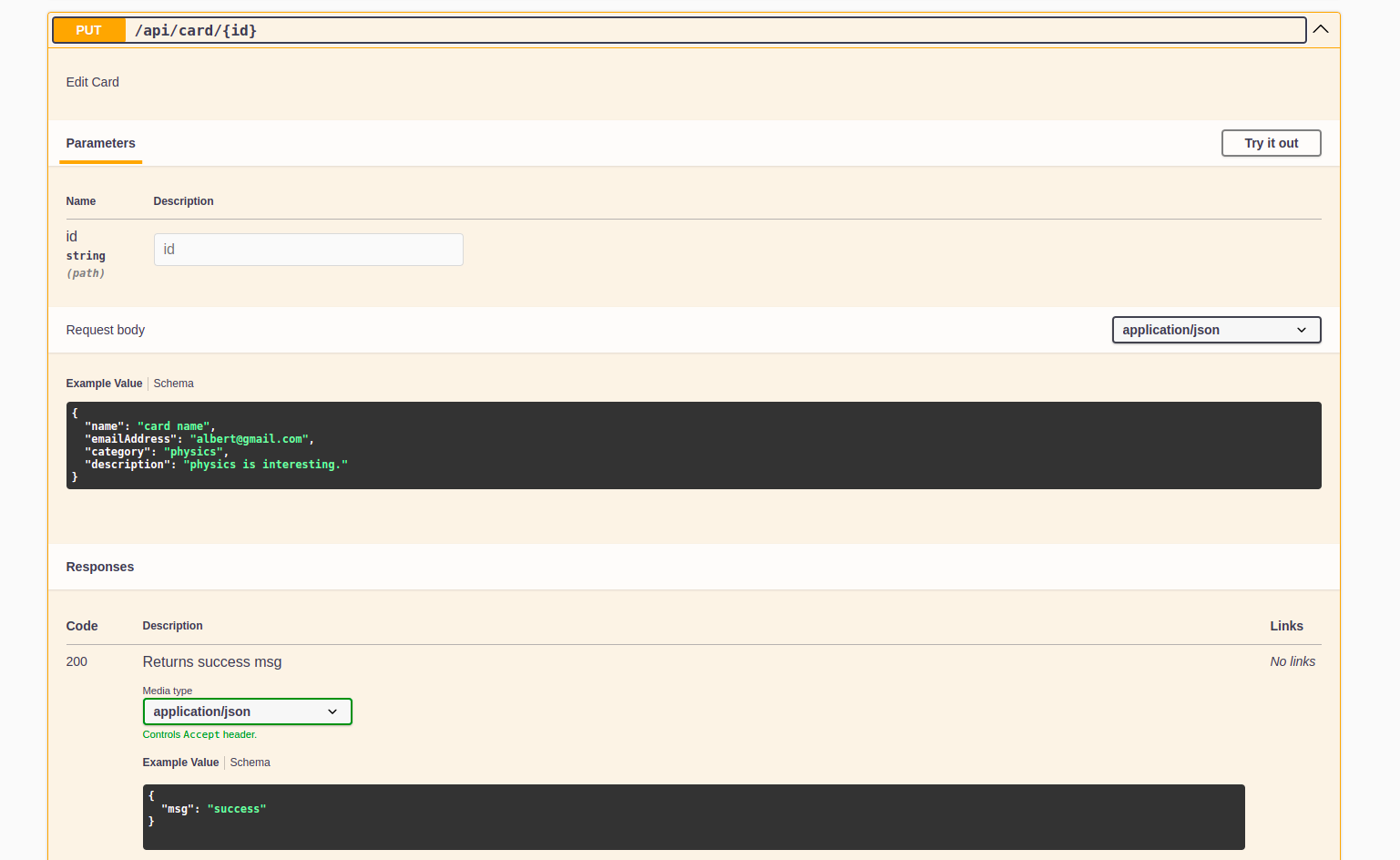
* Validate whether the Card exists
* Request Body validation
* Email Address Validation
* Check whether the Author Exists by the provided email address
* Check whether the author is authorized to perform card edit operation:
  + By checking whether the requested author created the card

(Email Address is used for this purpose)

* Edit Card and its Content
* Maintain Card and Content History

Response:

* Success Message for success case
* Error Message for the failure case

****

1. **DELETE: /api/card/{id}**

Delete Card by Card Id

//the status of the card is set as DELETED when the card is deleted.

Request Parameter:

* id // cardId

Request Body:

{

emailAddress

}

API method handles:

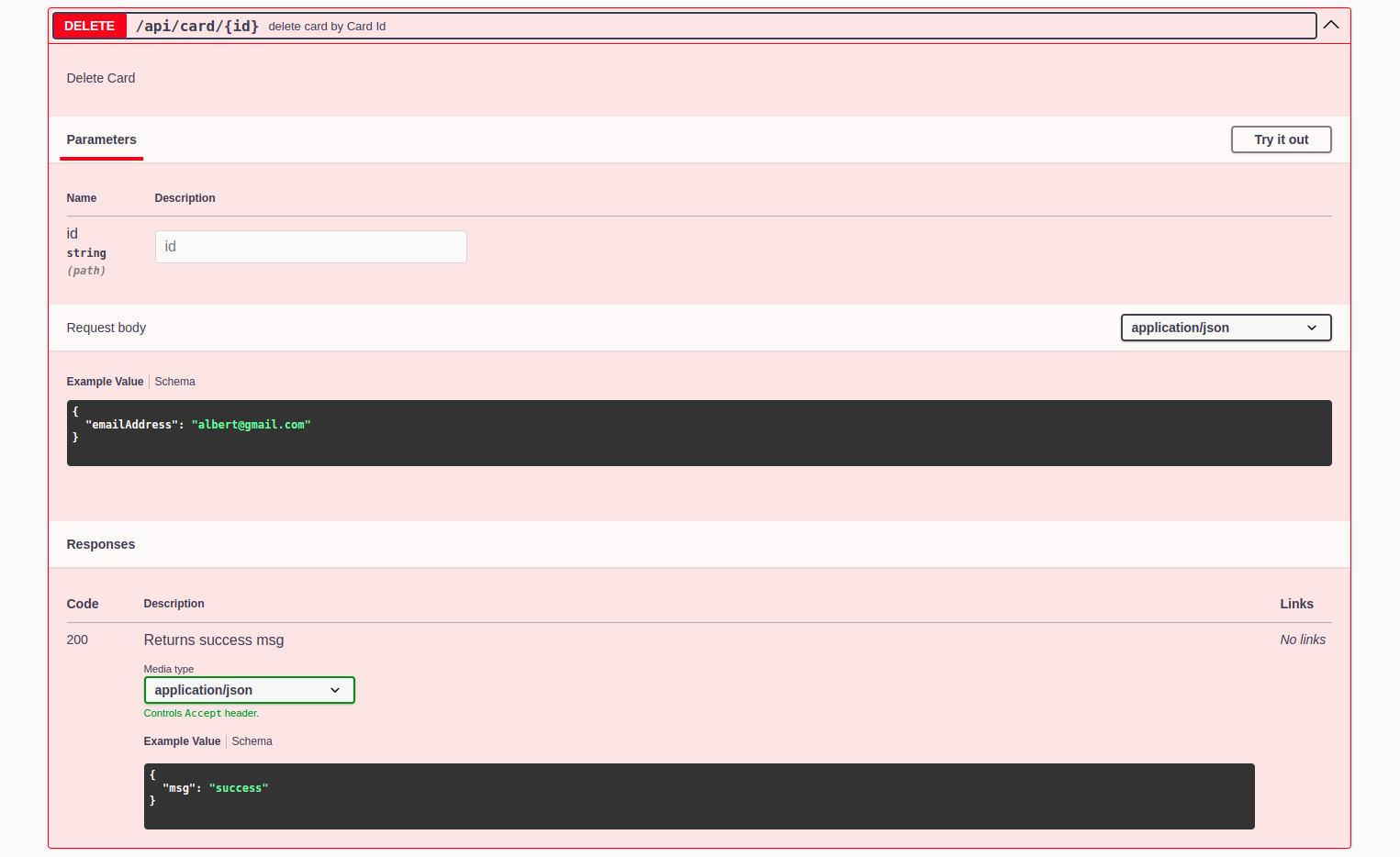
* Validate whether the Card exists
* Email Address Validation
* Check whether the Author Exists by the provided email address
* Check whether the author is authorized to perform card delete operation:
  + By checking whether the requested author created the card

(Email Address is used for this purpose)

* Delete Card and its Content
* Maintain Card and Content History

Response:

* Success Message for success case
* Error Message for the failure case



1. **GET: /api/card/{id}**

Fetch card details by Card Id

Request Parameter:

* id // cardId

API method handles:

* Validate whether the Card exists
* Fetch Card, its Content, and Author

Response:

* Response Body for success case

{

id,

name,

status,

author,

category,

description

}

- Error Message for failure case

