

API Testings

Authentication API's

Overview:

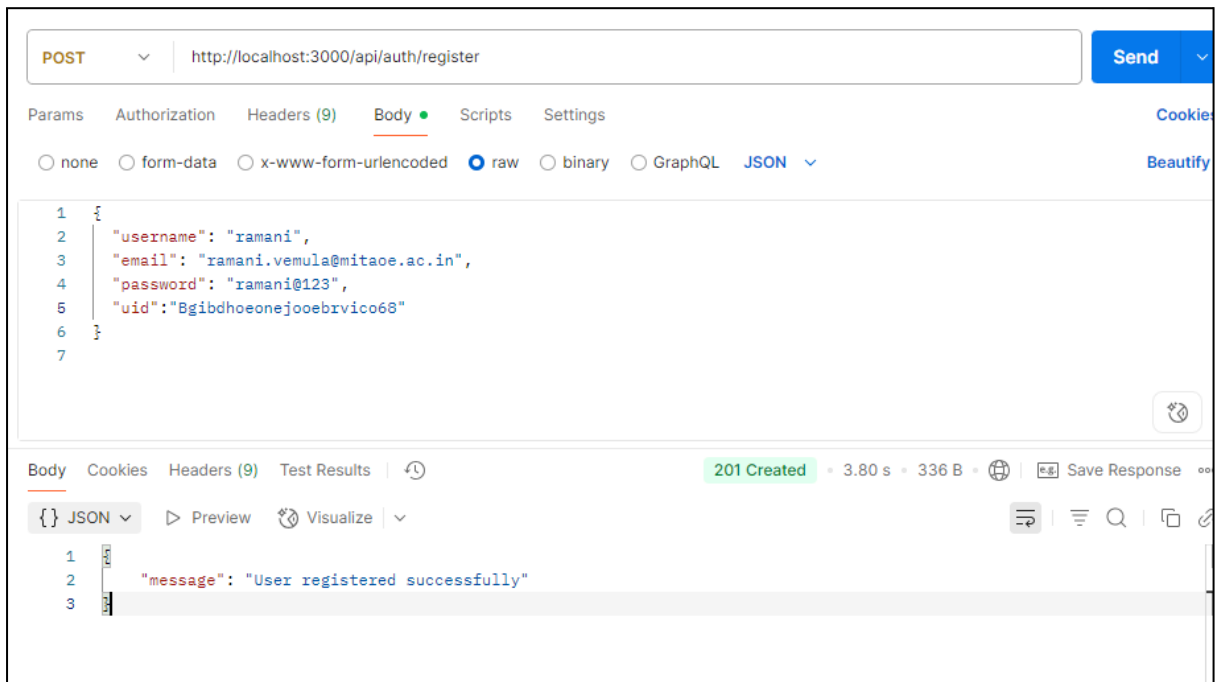
This API provides authentication features including user registration, login, OTP verification, password reset, and Google OAuth support.

Basic URL: <http://localhost:3000/api/auth>

1. Register User through entering details

URL: **POST** <http://localhost:3000/api/auth/register>

Postman:



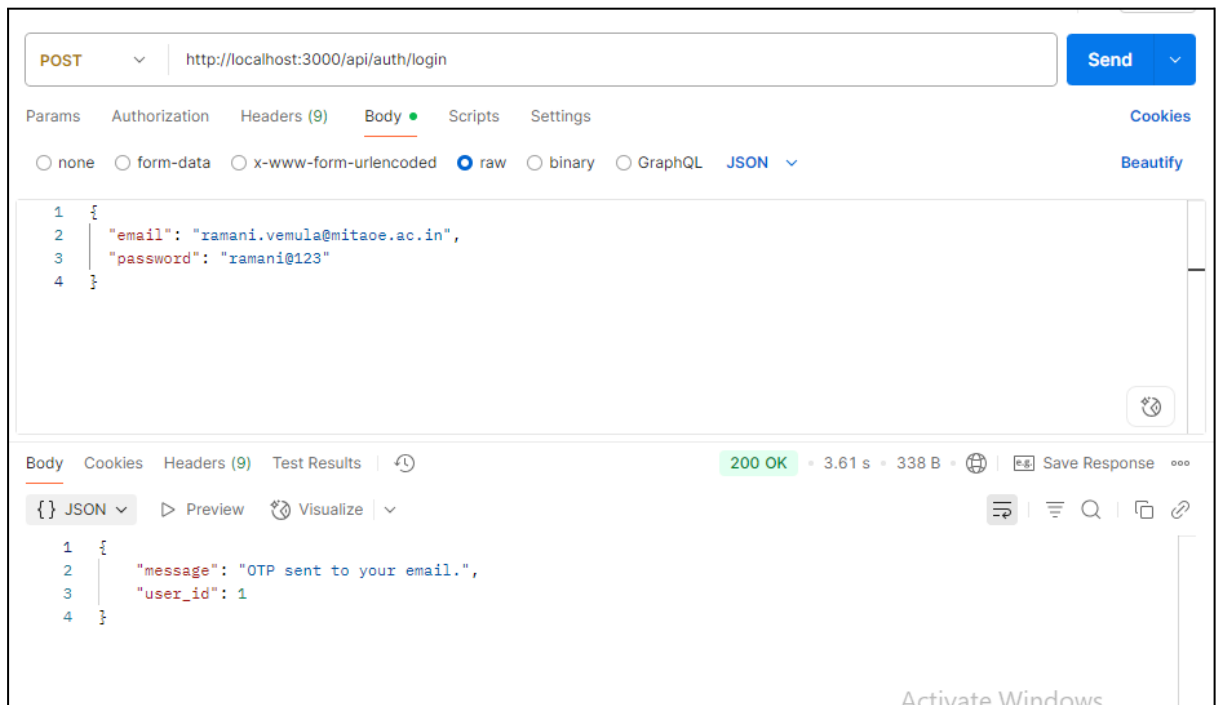
Database:

[illegible]

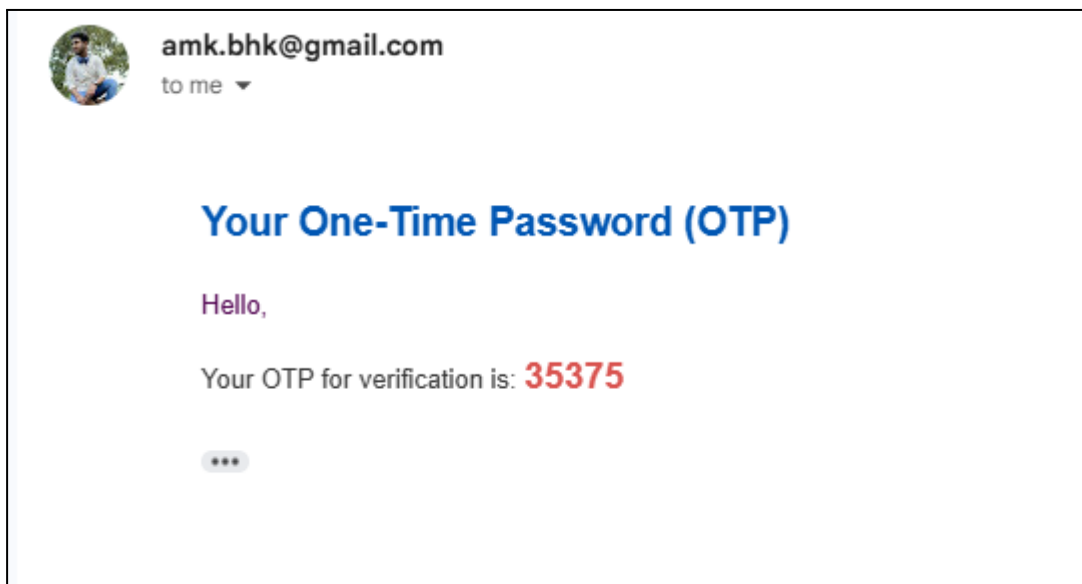
2. Login User through entering details

URL: **POST** <http://localhost:3000/api/auth/login>

Postman:



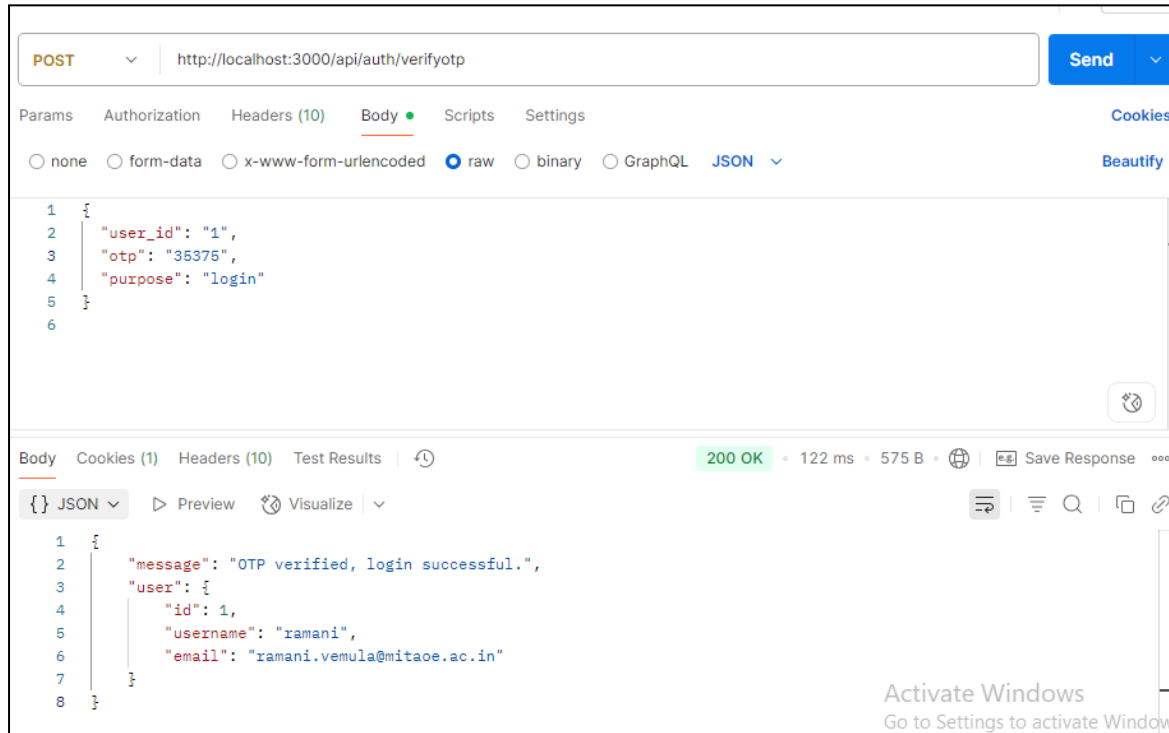
OTP will sent to email for verification:



3. Verifying account by entering the otp sent to mail:

URL: **POST** <http://localhost:3000/api/auth/verifyotp>

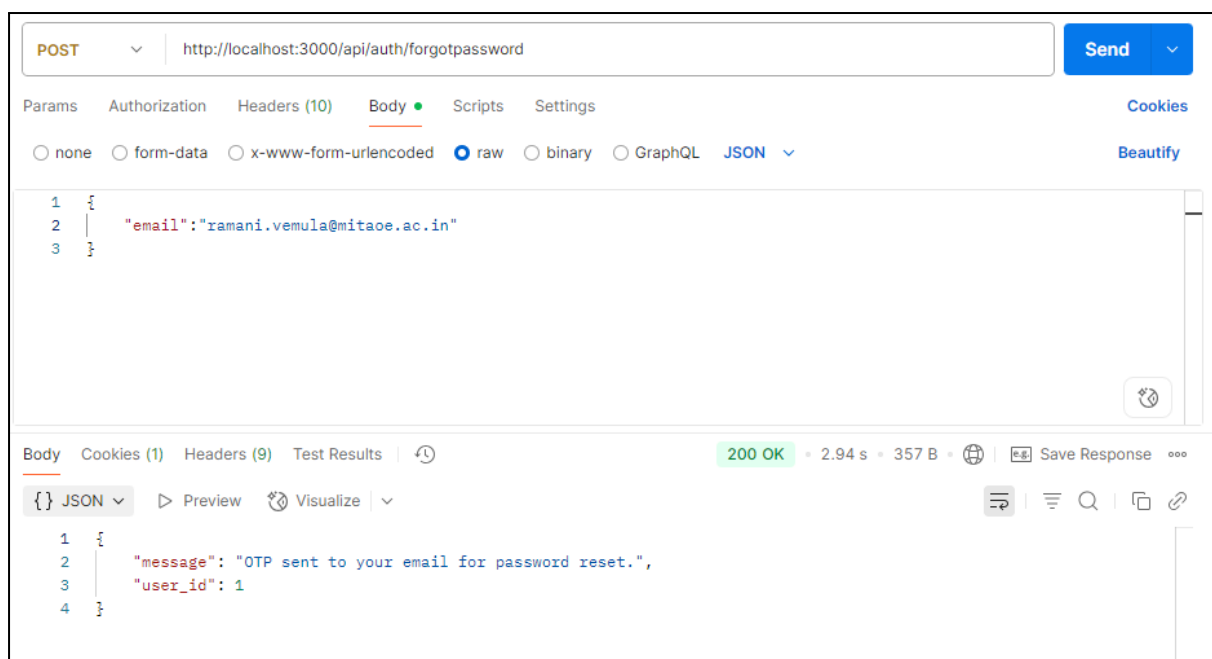
Postman:



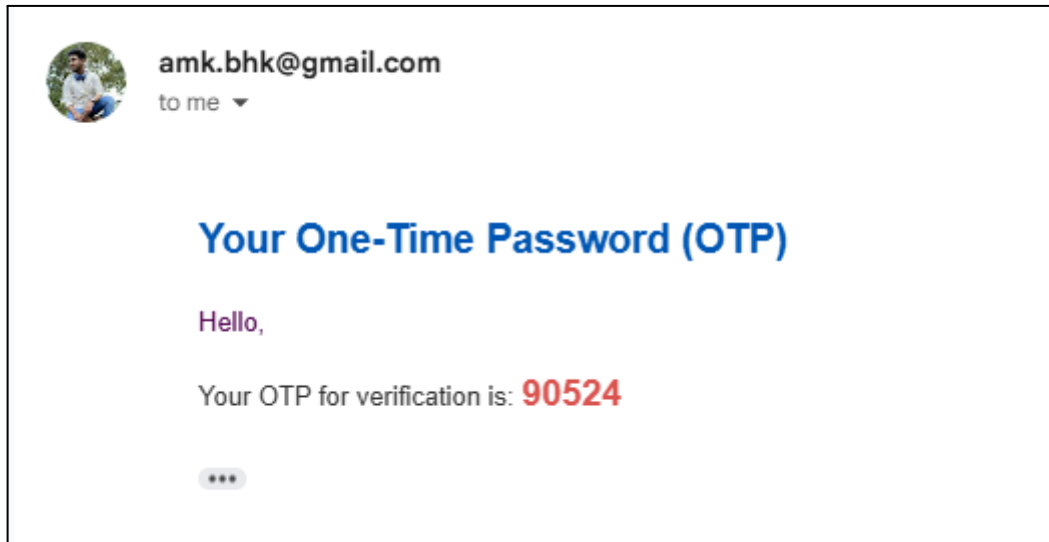
4. Forget Password

URL: **POST** <http://localhost:3000/api/auth/forgetpassword>

Postman:



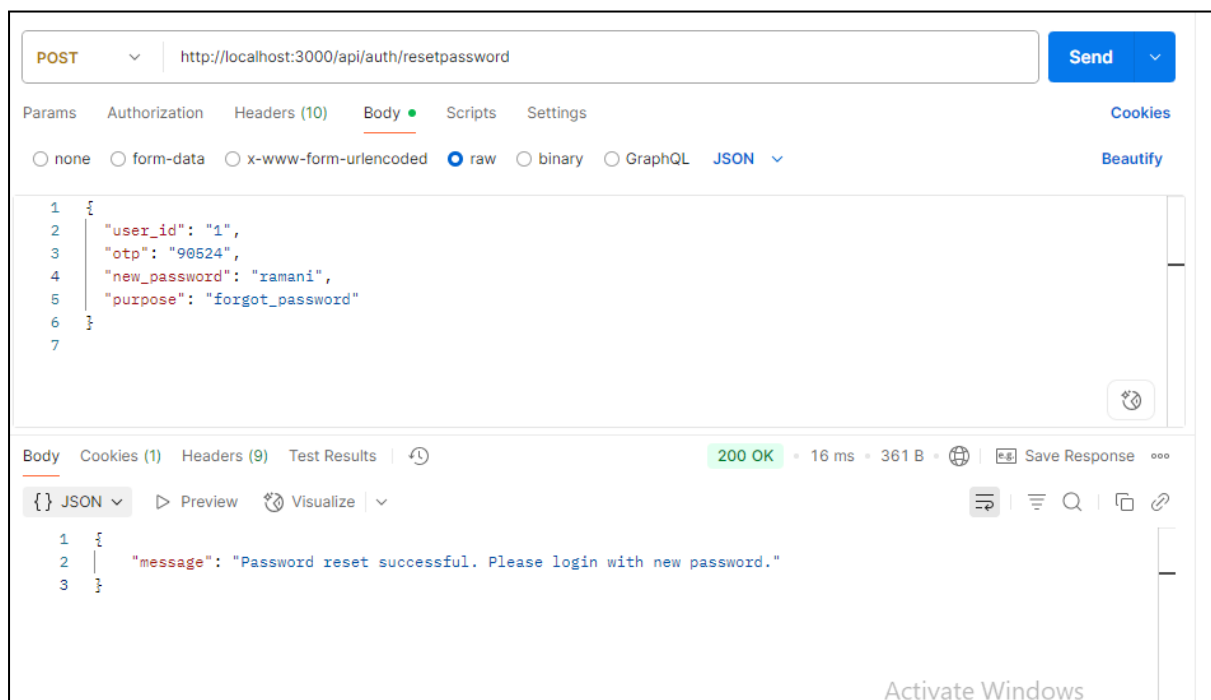
OTP will sent to email for verification:



5. Reset the password by entering OTP & New password

URL: **POST** <http://localhost:3000/api/auth/resetpassword>

Postman:

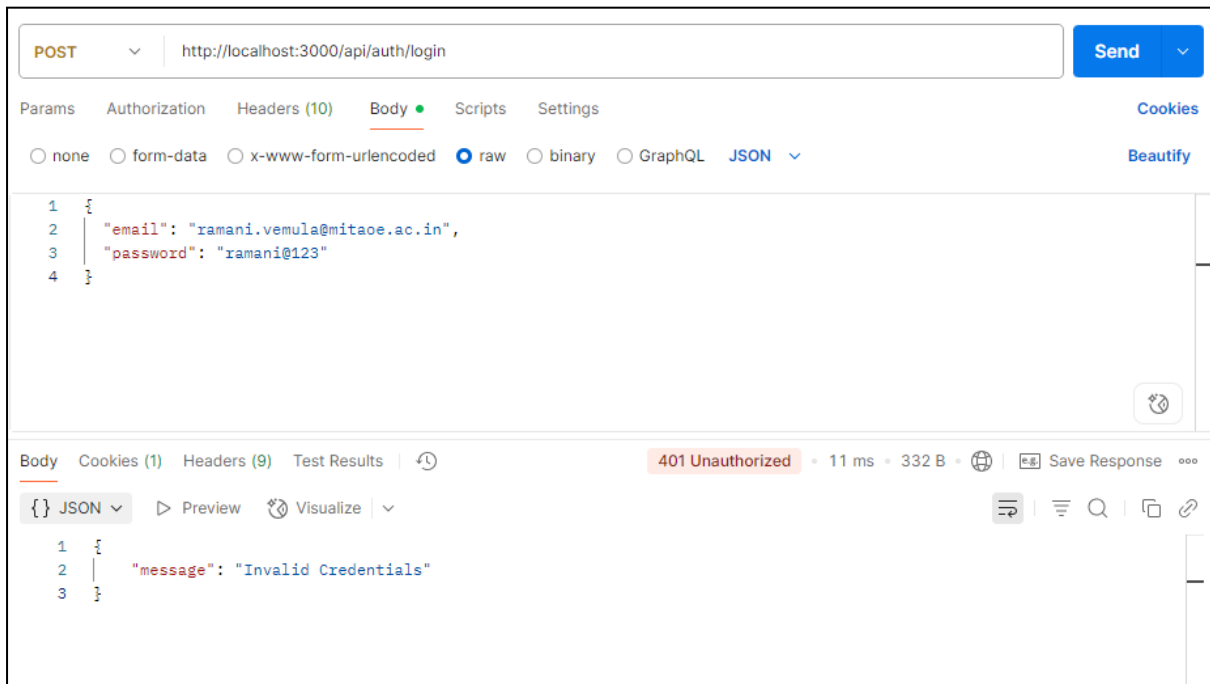


6. Login with previous and new password

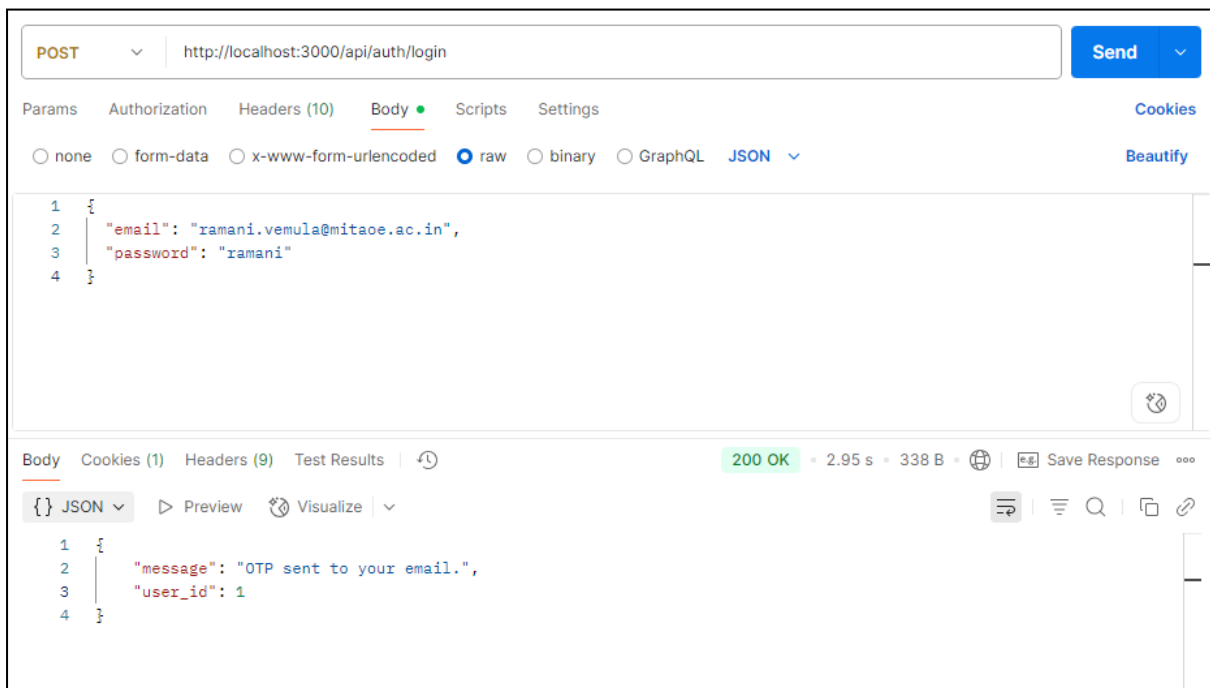
URL: **POST** <http://localhost:3000/api/auth/login>

Postman:

Old Password



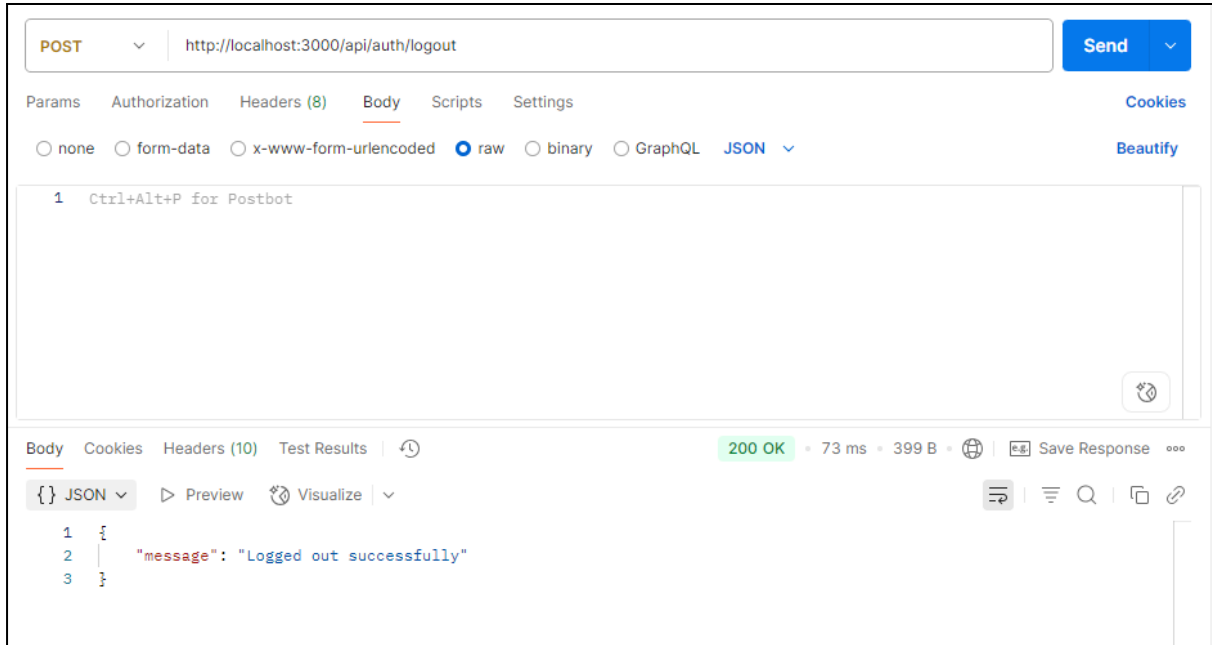
New password



7. Logout

URL: **POST** <http://localhost:3000/api/auth/logout>

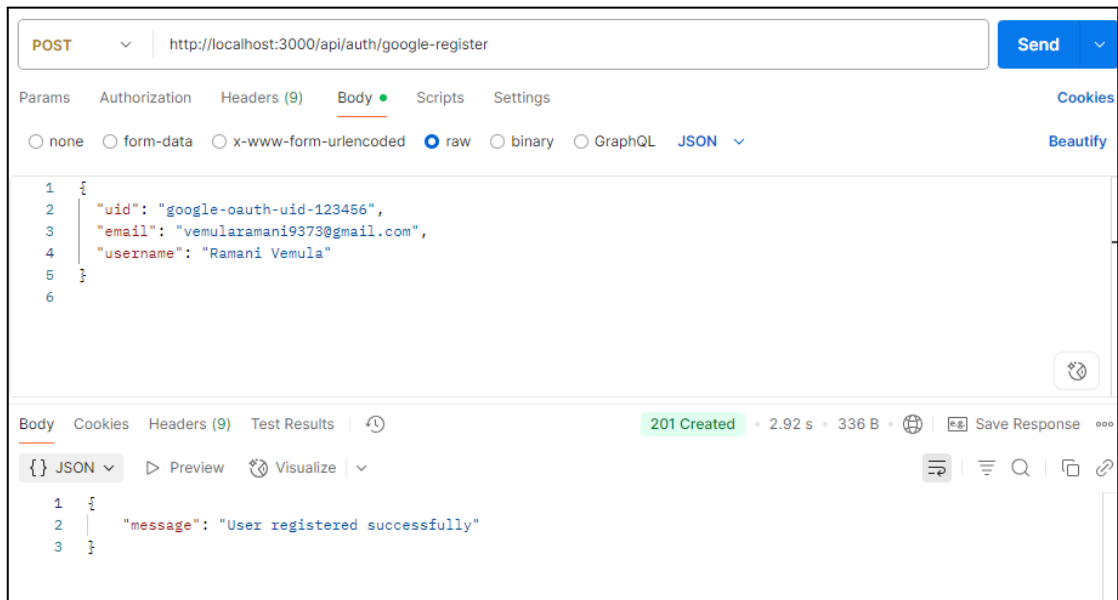
Postman:



8. Registering with the google

URL: **POST** <http://localhost:3000/api/auth/google-register>

Postman:



Database Schema:

[illegible]

9. Logging with the google

URL: **POST** <http://localhost:3000/api/auth/google-login>

Postman:

The image shows the Postman interface for a POST request to `http://localhost:3000/api/auth/google-login`. The request body is a JSON object with the following data:

```
1 {
2   "uid": "google-oauth-uid-123456",
3   "email": "vemularamani9373@gmail.com"
4 }
```

The response status is **200 OK** with a response time of 2.95 s and a body size of 338 B. The response body is a JSON object:

```
1 {
2   "message": "OTP sent to your email.",
3   "user_id": 2
4 }
```

The image shows an email interface for `amk.bhk@gmail.com`. The email is addressed "to me". The subject is "Your One-Time Password (OTP)". The body of the email contains the following text:

Hello,

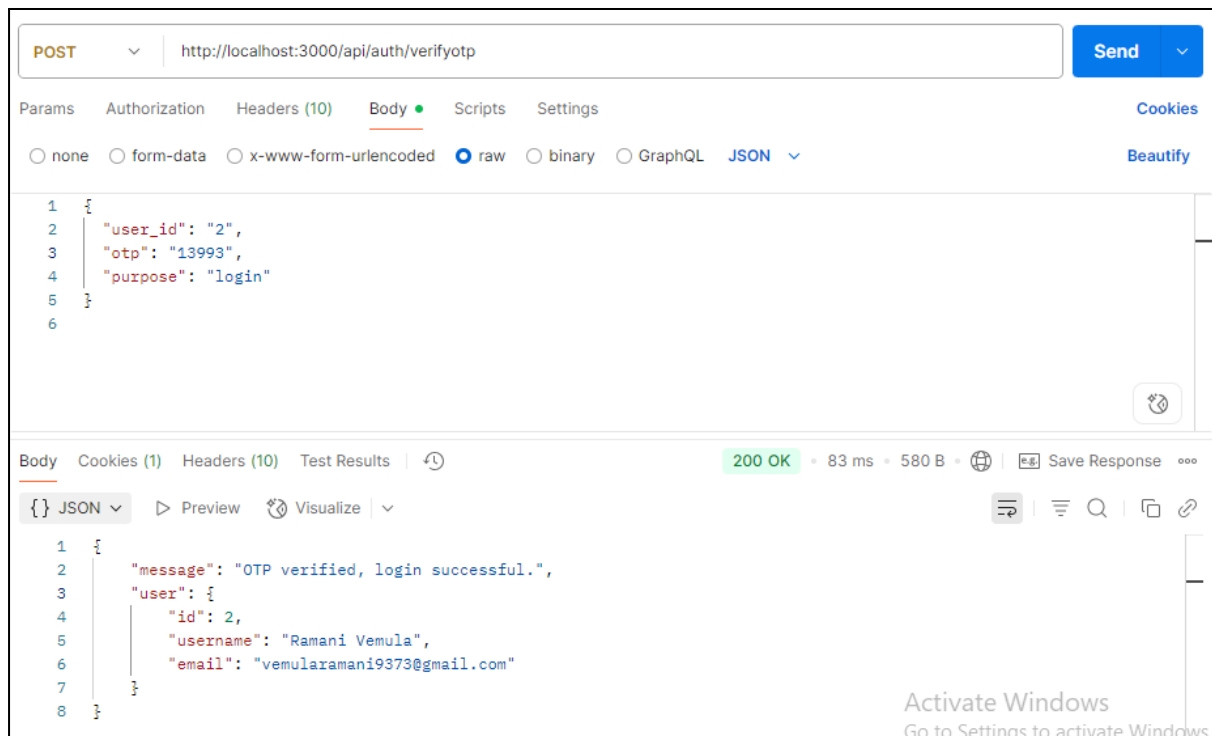
Your OTP for verification is: **13993**

Below the OTP, there is a button with three dots (⋮).

10.OTP Verification

URL: **POST** <http://localhost:3000/api/auth/verifyotp>

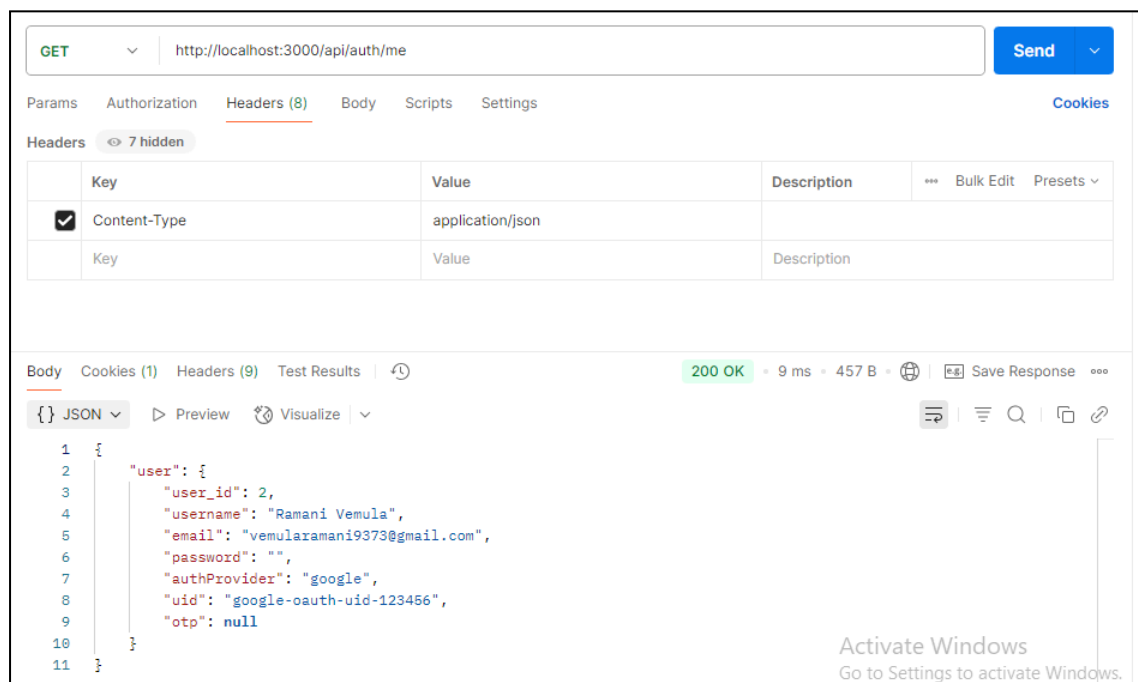
Postman:



11. Getting User Details

URL: **GET** <http://localhost:3000/api/auth/me>

Postman:



Model API's

Overview:

Models allow users to define and manage custom database structures by specifying a name and fields. The API supports creating, updating, viewing, and deleting models, and it automatically generates Sequelize code and stores metadata for each model.

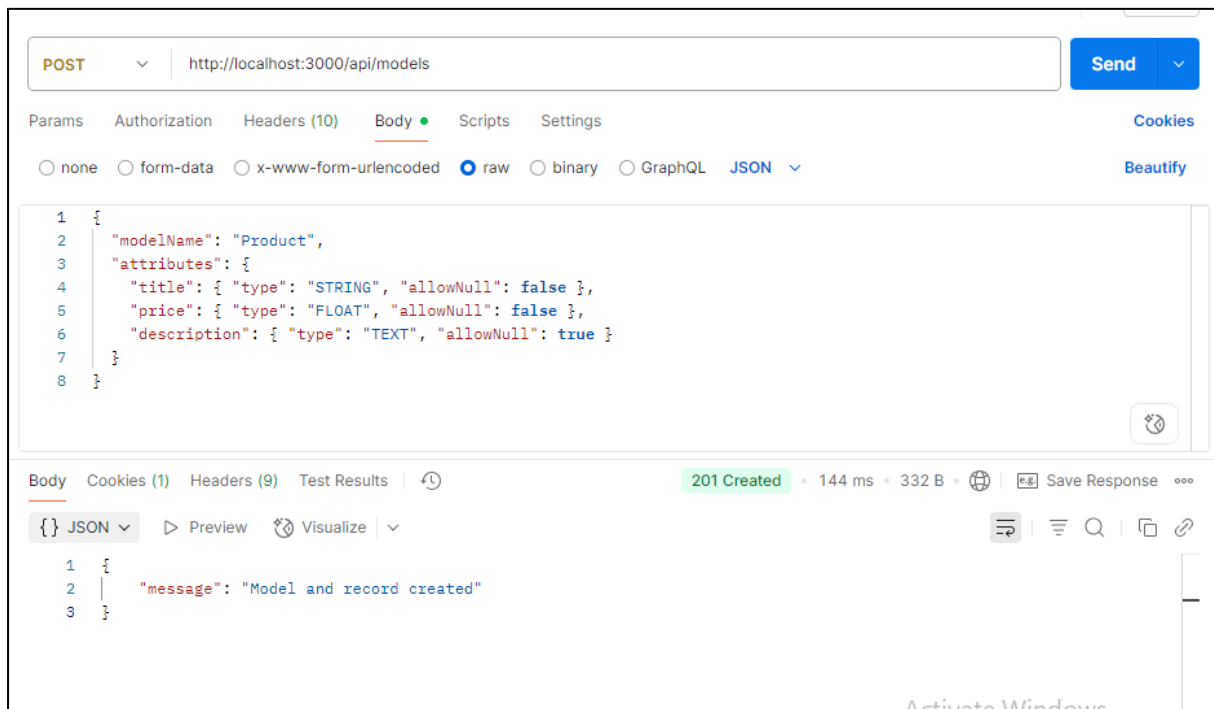
Basic URL: <http://localhost:3000/api/models>

1. Creating the model

URL: **POST** <http://localhost:3000/api/models>

Postman:

1. Product



Database

Result Grid					
Filter Rows:					
	id	name	code	metadata	user_id
	1	Product	<pre>const { DataTypes } = require('sequelize'); const sequelize = require('../config/db'); const Product = sequelize.define('Product', { title: { type: DataTypes.STRING, allowNull: false }, price: { type: DataTypes.FLOAT, allowNull: false }, description: {</pre>	<pre>{ "attributes": { "price": { "type": "FLOAT", "allowNull": false }, "title": { "type": "STRING", "allowNull": false }, "description": { "type": "TEXT", "allowNull": true } } }</pre>	1

2. Book

POST http://localhost:3000/api/models

Params Authorization Headers (10) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "modelName": "Book",
3   "attributes": {
4     "title": { "type": "STRING", "allowNull": false },
5     "author": { "type": "STRING", "allowNull": false },
6     "publishedYear": { "type": "INTEGER", "allowNull": true },
7     "genre": { "type": "STRING", "allowNull": true }
8   }
9 }
10
```

201 Created • 16 ms • 332 B

Body Cookies (1) Headers (9) Test Results

```
1 {
2   "message": "Model and record created"
3 }
```

Database

id	name	code	metadata	user_id
2	Book	<pre>const { DataTypes } = require('sequelize'); const sequelize = require('../config/db'); const Book = sequelize.define('Book', { title: { type: DataTypes.STRING, allowNull: false }, author: { type: DataTypes.STRING, allowNull: true }, publishedYear: { type: DataTypes.INTEGER, allowNull: true }, genre: { type: DataTypes.STRING, allowNull: true } });</pre>	<pre>{ "attributes": { "genre": { "type": "STRING", "allowNull": true }, "title": { "type": "STRING", "allowNull": false }, "author": { "type": "STRING", "allowNull": false }, "publishedYear": { "type": "INTEGER", "allowNull": true } } }</pre>	1
NULL	NULL	NULL	NULL	NULL

3. Student

POST http://localhost:3000/api/models

Params Authorization Headers (10) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "modelName": "student",
3   "attributes": {
4     "name": { "type": "STRING", "allowNull": false },
5     "branch": { "type": "STRING", "allowNull": false },
6     "gender": { "type": "STRING", "allowNull": false },
7     "year": { "type": "INTEGER", "allowNull": false }
8   }
9 }
10
```

201 Created • 10 ms • 332 B

Body Cookies (1) Headers (9) Test Results

```
1 {
2   "message": "Model and record created"
3 }
```

Database

	id	name	code	metadata	user_id
	1	Product	const { DataTypes } = require('sequelize'); const sequelize = r...	{attributes: {price: {type: 'FLOAT', 'allow...	1
	2	Book	const { DataTypes } = require('sequelize'); const sequelize = r...	{attributes: {Pages: {type: 'INTEGER', 'all...	1
▶	4	student	const { DataTypes } = require('sequelize'); const sequelize = r...	{attributes: {name: {type: 'STRING', 'allo...	1
*	NULL	NULL	NULL	NULL	NULL

After updation in code:

POST http://localhost:3000/api/models

Send

Params Authorization Headers (10) Body Scripts Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2   "modelName": "student",
3   "attributes": {
4     "name": { "type": "STRING", "allowNull": false },
5     "branch": { "type": "STRING", "allowNull": false },
6     "gender": { "type": "STRING", "allowNull": false },
7     "year": { "type": "INTEGER", "allowNull": false }
8   }
9 }
10
```

Body Cookies (1) Headers (9) Test Results 201 Created 10 ms 332 B Save Response

JSON Preview Visualize

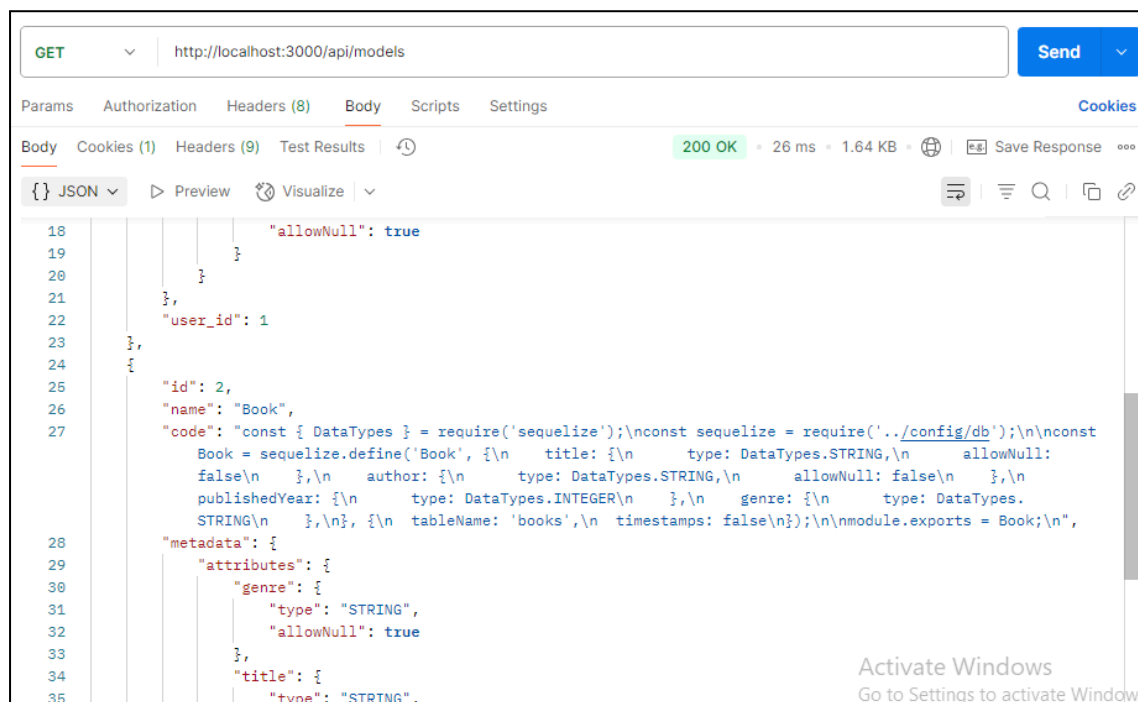
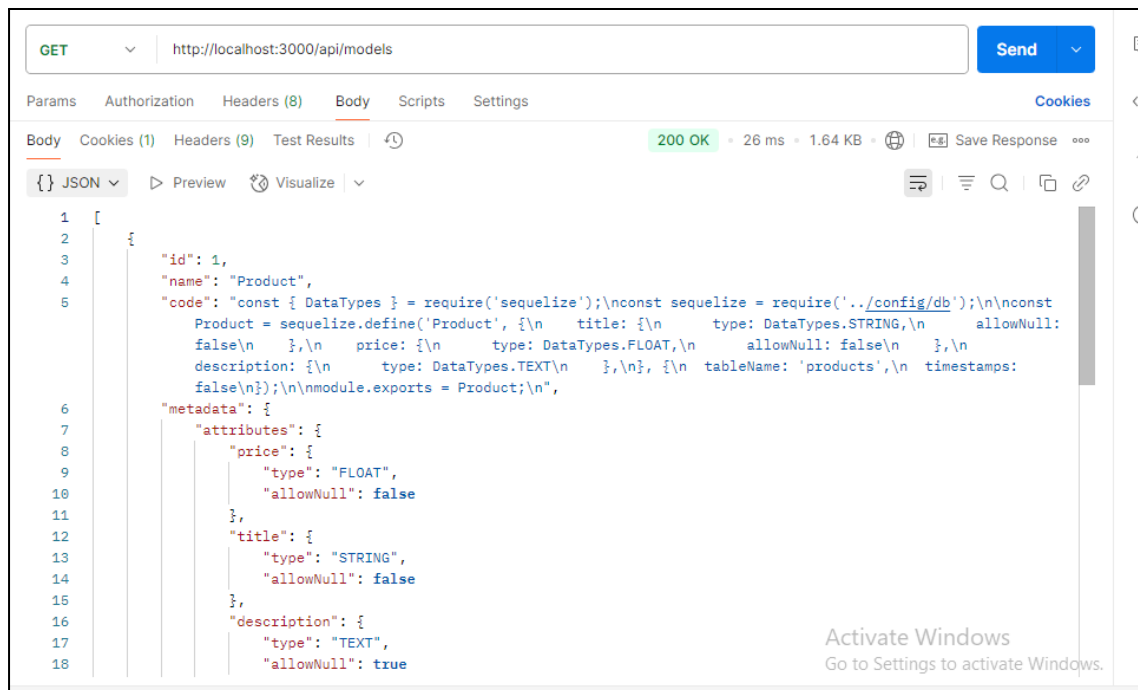
```
1 {
2   "message": "Model and record created"
3 }
```

Result Grid		Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
	id	name	code	metadata	user_id
	2	Book	<pre>const { DataTypes } = require('sequelize'); const sequelize = require('./config/db'); const Book = sequelize.define('Book', { title: { type: DataTypes.STRING, allowNull: false }, author: { type: DataTypes.STRING, allo...</pre>	<pre>{attributes: {Pages: {type: "INTEGER", "allowNull": true}, "title": {type: "STRING", "allowNull": false, "author": {type: "STRING", "allowNull": false}, "publishedYear": {type: "INTEGER", "allowNull": true}}}, "association": []}</pre>	1
▶	5	student	<pre>const { Model, DataTypes } = require('sequelize'); const sequelize = require('./config/db'); class student extends Model {} student.init({ name: { type: DataTypes.STRING, allowNull: false }, branch: { type: DataTypes.STRI...</pre>	<pre>{attributes: {name: {type: "STRING", "allowNull": false}, "year": {type: "INTEGER", "allowNull": false}, "branch": {type: "STRING", "allowNull": false}, "gender": {type: "STRING", "allowNull": false}}}</pre>	1
*	NULL	NULL	NULL	NULL	NULL

2. Getting all the models created by user_id : 1

URL: **GET** <http://localhost:3000/api/models>

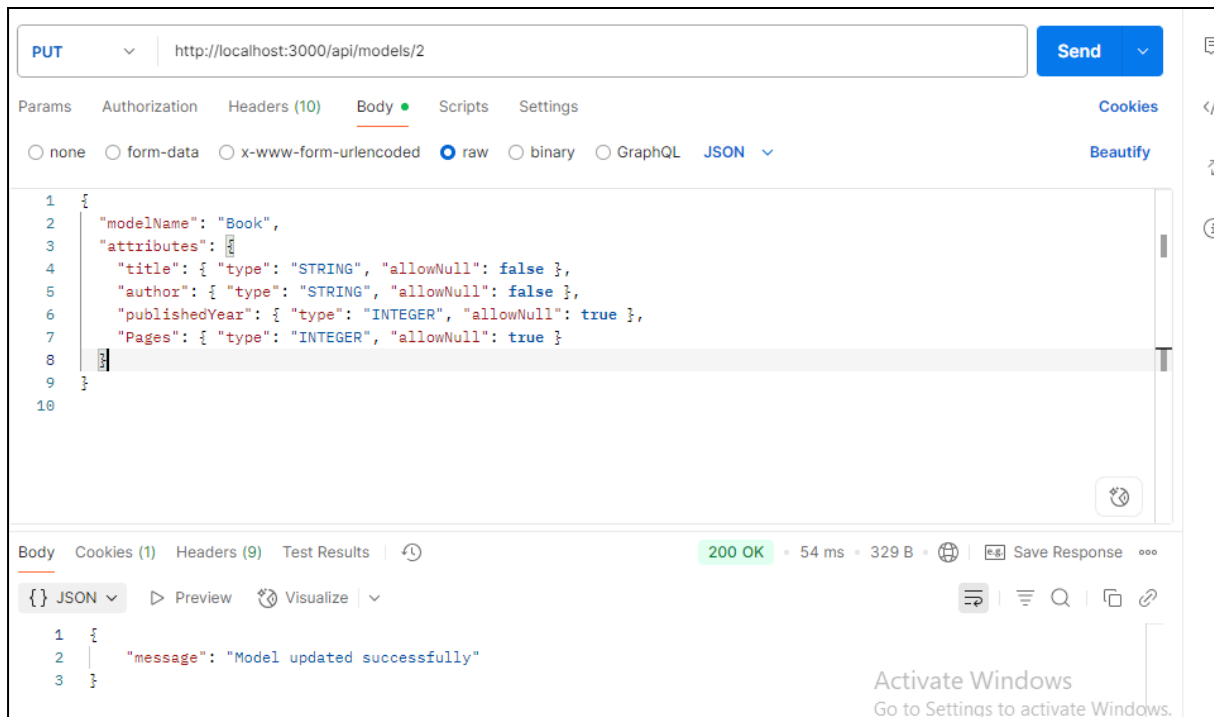
Postman:



3. Updating the existing model → 2 [Book]

URL: **PUT** <http://localhost:3000/api/models/2>

Postman:



Database

Result Grid					
Filter Rows:					
Edit: Export/Import: Wrap Cell Content:					
id	name	code	metadata	user_id	
2	Book	<pre>const { DataTypes } = require('sequelize'); const sequelize = require('../config/db'); const Book = sequelize.define('Book', { title: { type: DataTypes.STRING, allowNull: false }, author: { type: DataTypes.STRING, allowNull: false }, publishedYear: { type: DataTypes.INTEGER }, Pages: { type: DataTypes.INTEGER }, }, { tableName: 'books', });</pre>	<pre>{ "attributes": { "Pages": { "type": "INTEGER", "allowNull": true }, "title": { "type": "STRING", "allowNull": false }, "author": { "type": "STRING", "allowNull": false }, "publishedYear": { "type": "INTEGER", "allowNull": true } } }</pre>	1	

After updation in code:

Result Grid		Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
id	name	code	metadata	user_id	
5	student	<pre>const { Model, DataTypes } = require('sequelize'); const sequelize = require('../config/db'); class student extends Model {} student.init({ name: { type: DataTypes.STRING, allowNull: false }, branch: { type: DataTypes.STRING, allowNull: false }, gender: { type: DataTypes.STRING, allowNull: false }, year: { type: DataTypes.INTEGER, allowNull: false }, address: { type: DataTypes.STRING }, }, {</pre>	<pre>{ "attributes": { "name": { "type": "STRING", "allowNull": false, }, "year": { "type": "INTEGER", "allowNull": false, }, "branch": { "type": "STRING", "allowNull": false, }, "gender": { "type": "STRING", "allowNull": false, }, "address": { "type": "String", "allowNull": true } } }</pre>	1	

4. Deleting the existing model → 4 [Student]

URL: DELETE <http://localhost:3000/api/models/4>

Postman:

DELETE

▼

http://localhost:3000/api/models/4

Send

▼

Params

Authorization

Headers (8)

Body

Scripts

Settings

Cookies

☒ none

☐ form-data

☐ x-www-form-urlencoded

☐ raw

☐ binary

☐ GraphQL

This request does not have a body

Body

Cookies (1)

Headers (9)

Test Results

🔄

200 OK

• 13 ms

• 316 B

🌐

📄 Save Response

⋮

{ } JSON

▼

▶ Preview

🔗 Visualize

▼

```
1 {
2   "message": "Model deleted"
3 }
```

Database

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Content:

	id	name	code	metadata	user_id
▶	1	Product	const { DataTypes } = require('sequelize'); const sequelize = r...	{ "attributes": { "price": { "type": "FLOAT", "allow...	1
	2	Book	const { DataTypes } = require('sequelize'); const sequelize = r...	{ "attributes": { "Pages": { "type": "INTEGER", "all...	1
*	NULL	NULL	NULL	NULL	NULL

RELATIONSHIP API's

Overview:

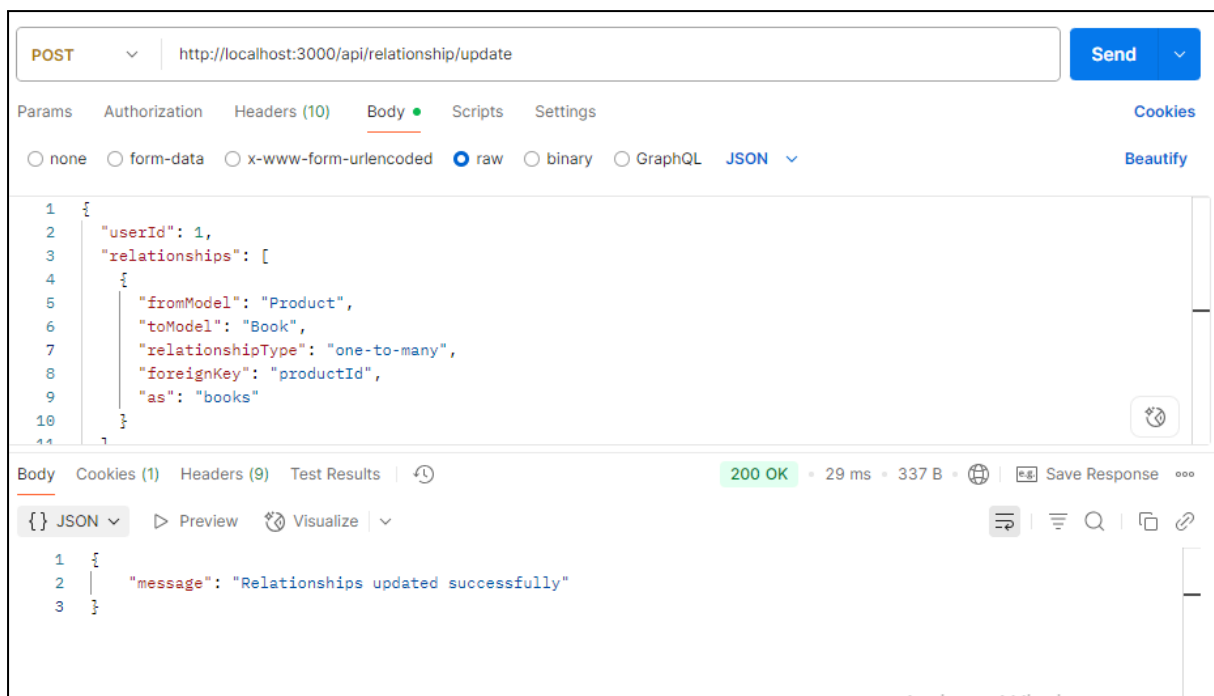
This module provides APIs to define, update, and delete associations between two models. The system ensures that both forward and reverse associations are stored and reflected in the Sequelize code and metadata, making models fully relational and manageable through the UI or programmatically.

Basic URL: <http://localhost:3000/api/relationship>

1. Updating the code by adding the relation between Product & book→ one-to-many

URL: **POST** <http://localhost:3000/api/relationship/update>

Postman:



Database → association added [hasMany]

Result Grid					Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
id	name	code	metadata	user				
1	Product	<pre>const { DataTypes } = require('sequelize'); const sequelize = require('../config/db'); const Product = sequelize.define('Product', { productId: { type: DataTypes.INTEGER }, title: { type: DataTypes.STRING, allowNull: false }, ... });</pre>	<pre>{ "attributes": { "price": { "type": "FLOAT", "allowNull": false }, "title": { "type": "STRING", "allowNull": false }, "productId": { "type": "INTEGER", "allowNull": true }, "description": { "type": "TEXT", "allowNull": true }, "association": { "as": "books", "type": "hasMany", "target": "Book", "foreignKey": "productId" } } }</pre>	1				
2	Book	<pre>const { DataTypes } = require('sequelize'); const sequelize = require('../config/db'); const Book = sequelize.define('Book', { title: { type: DataTypes.STRING, allowNull: false }, author: { type: DataTypes.STRING, all...</pre>	<pre>{ "attributes": { "Pages": { "type": "INTEGER", "allowNull": true }, "title": { "type": "STRING", "allowNull": false }, "author": { "type": "STRING", "allowNull": false }, "publishedYear": { "type": "INTEGER", "allowNull": true }, "association": { "as": "reverse_...</pre>	1				
NULL	NULL	NULL	NULL	NULL				

2. Updating the code by another relation between Product & book → many-to-many

URL: **POST** <http://localhost:3000/api/relationship/update>

Postman:

POST <http://localhost:3000/api/relationship/update> Send

Params Authorization Headers (10) Body Scripts Settings Cookies Beautify

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON

```
1 {
2   "userId": 1,
3   "relationships": [
4     {
5       "fromModel": "Product",
6       "toModel": "Book",
7       "relationshipType": "many-to-many",
8       "foreignKey": "productId",
9       "as": "books"
10    }
11  ]
12 }
```

Body Cookies (1) Headers (9) Test Results 200 OK • 36 ms • 337 B Save Response

{ } JSON Preview Visualize

```
1 {
2   "message": "Relationships updated successfully"
3 }
```

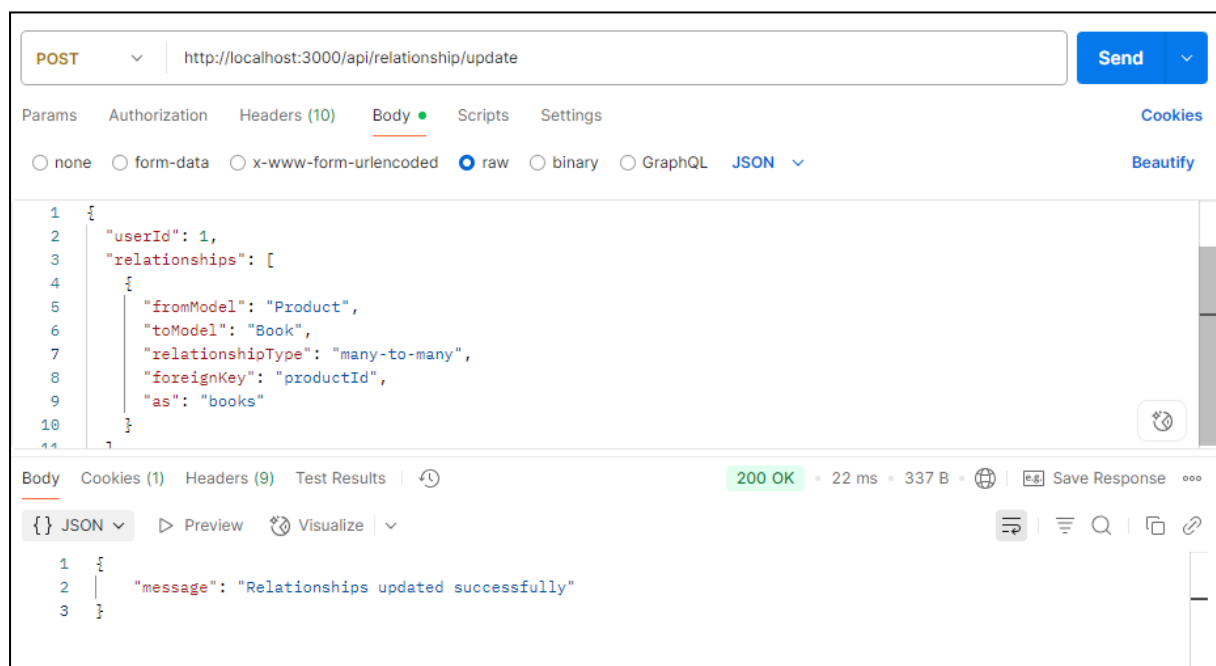
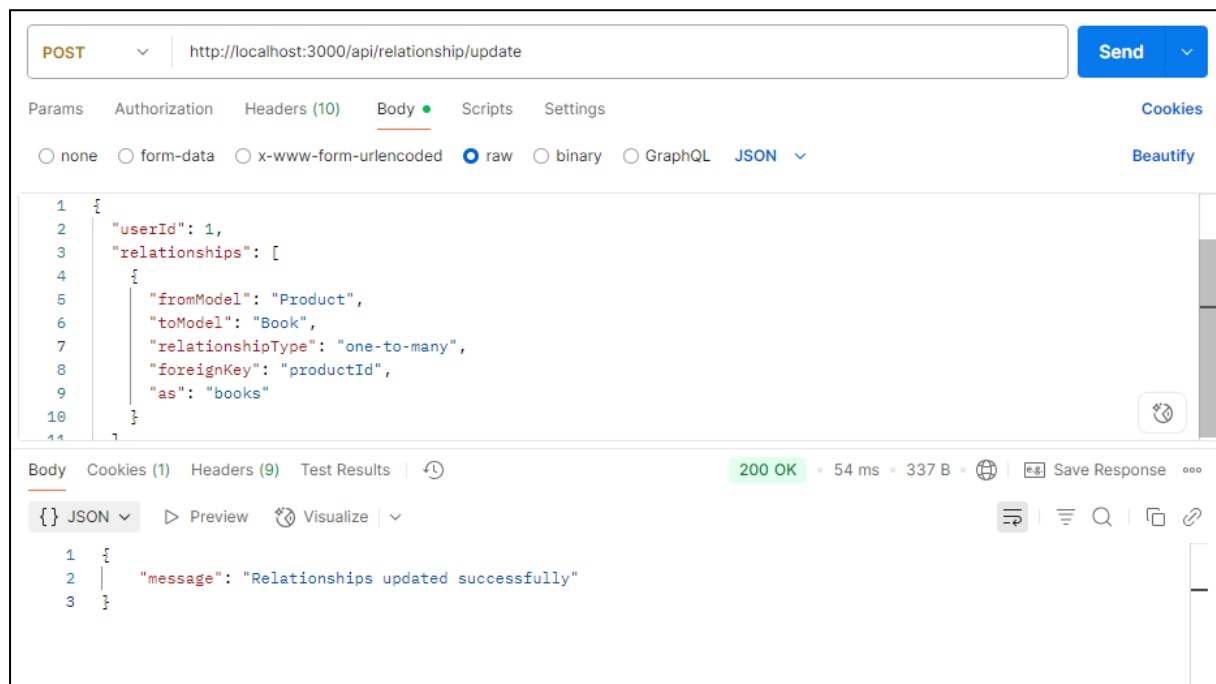

id	name	code	metadata	user_id
1	Product	<pre>const { DataTypes } = require('sequelize'); const sequelize = require('../config/db'); const Product = sequelize.define('Product', { productId: { type: DataTypes.INTEGER }, title: { type: DataTypes.STRING, allowNull: false }, ... });</pre>	<pre>{ "attributes": { "price": { "type": "FLOAT", "allowNull": false }, "title": { "type": "STRING", "allowNull": false }, "productId": { "type": "INTEGER", "allowNull": true }, "description": { "type": "TEXT", "allowNull": true }, "association": [{ "as": "books", "type": "hasMany", "target": "Book", "foreignKey": "productId", "as": "books", "type": "belongsToMany", "target": "Book", "foreignKey": "productId" }] } }</pre>	1
2	Book	<pre>const { DataTypes } = require('sequelize'); const sequelize = require('../config/db'); const Book = sequelize.define('Book', { title: { type: DataTypes.STRING, allowNull: false }, author: { type: DataTypes.STRING, allowNull: false }, ... });</pre>	<pre>{ "attributes": { "Pages": { "type": "INTEGER", "allowNull": true }, "title": { "type": "STRING", "allowNull": false }, "author": { "type": "STRING", "allowNull": false }, "publishedYear": { "type": "INTEGER", "allowNull": true }, "association": [{ "as": "reverse_..." }] } }</pre>	1
NULL	NULL	NULL	NULL	NULL

ISSUE: Duplicate Associations in Metadata When Updating Relationships

→ Instead of replacing the existing one-to-many relationship, the system **added** a new many-to-many relationship

After Correcting the Issue:

id	name	code	metadata	user_id
1	Product	<pre>const { DataTypes } = require('sequelize'); const sequelize = require('../config/db'); const Product = sequelize.define('Product', { productId: { type: DataTypes.INTEGER }, title: { type: DataTypes.STRING, allowNull: false }, ... });</pre>	<pre>{ "attributes": { "price": { "type": "FLOAT", "allowNull": false }, "title": { "type": "STRING", "allowNull": false }, "productId": { "type": "INTEGER", "allowNull": true }, "description": { "type": "TEXT", "allowNull": true }, "association": [{ "as": "books", "type": "hasMany", "target": "Book", "foreignKey": "productId" }] } }</pre>	1
2	Book	<pre>const { DataTypes } = require('sequelize'); const sequelize = require('../config/db'); const Book = sequelize.define('Book', { title: { type: DataTypes.STRING, allowNull: false }, author: { type: DataTypes.STRING, allowNull: false }, ... });</pre>	<pre>{ "attributes": { "Pages": { "type": "INTEGER", "allowNull": true }, "title": { "type": "STRING", "allowNull": false }, "author": { "type": "STRING", "allowNull": false }, "publishedYear": { "type": "INTEGER", "allowNull": true }, "association": [{ "as": "reverse_..." }] } }</pre>	1
5	student	<pre>const { Model, DataTypes } = require('sequelize'); const sequelize = require('../config/db'); class student extends Model {} student.init({</pre>	<pre>{ "attributes": { "name": { "type": "STRING", "allowNull": false }, "year": { "type": "INTEGER", "allowNull": false }, "branch": { "type": "STRING", "allowNull": false }, "gender": { "type": "STRING", "allowNull": false }, "address": { "type": "String", "allowNull": ... } } }</pre>	1



3. Delete the relation between Product & book → one-to-many

URL: **POST** <http://localhost:3000/api/relationship/delete>

Postman:

POST http://localhost:3000/api/relationship/delete

Params Authorization Headers (10) Body Scripts Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1 {
2   "userId": 1,
3   "fromModel": "Product",
4   "toModel": "Book",
5   "relationshipType": "one-to-many",
6   "foreignKey": "productId",
7   "as": "books"
8 }
9

```

Body Cookies (1) Headers (9) Test Results 200 OK 34 ms 1.94 KB Save Response

JSON Preview Visualize

```

1 {
2   "message": "Relationship deleted successfully (both directions)",
3   "forward": {
4     "code": "const { DataTypes } = require('sequelize');\nconst sequelize = require('../config/db');\nconst
Product = sequelize.define('Product', {\n  productId: {\n    type: DataTypes.INTEGER\n  },\n
title: {\n    type: DataTypes.STRING,\n    allowNull: false\n  },\n  price: {\n    type:
DataTypes.FLOAT,\n    allowNull: false\n  },\n  description: {\n    type: DataTypes
TEXT\n  },\n}, {\n  tableName: 'products',\n  timestamps: false\n});\nmodule.exports = Product;\n"

```

Database → relation deleted successfully from both side

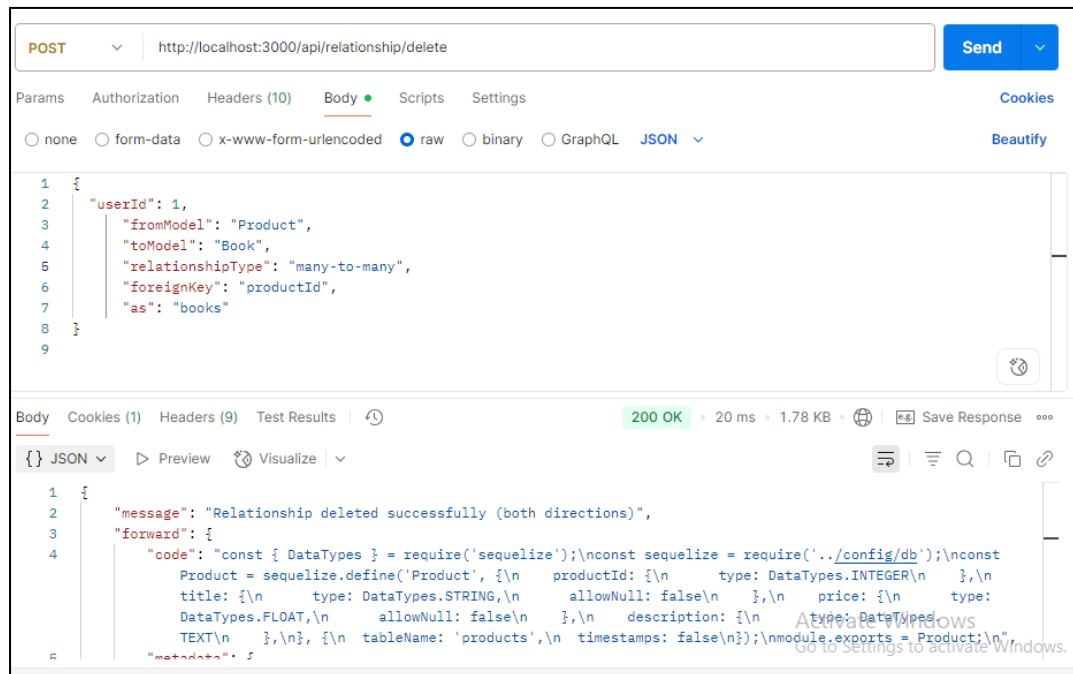
id	name	code	metadata	user_id
2	Book	const { DataTypes } = require('sequelize'); const sequelize = require('../config/db'); const Book = sequelize.define('Book', { title: { type: DataTypes.STRING, allowNull: false }, author: { type: DataTypes.STRING, allo...	{ "attributes": { "Pages": { "type": "INTEGER", "allowNull": true, "title": { "type": "STRING", "allowNull": false, "author": { "type": "STRING", "allowNull": false, "publishedYear": { "type": "INTEGER", "allowNull": true, "association": { [{"as": "reverse_books", "type": "belongsToMany", "target": "Product", "foreignKey": "productId"}] } } } } } } }	1
1	Product	const { DataTypes } = require('sequelize'); const sequelize = require('../config/db'); const Product = sequelize.define('Product', { productId: { type: DataTypes.INTEGER }, title: { type: DataTypes.STRING, allowNull: false }, ...	{ "attributes": { "price": { "type": "FLOAT", "allowNull": false, "title": { "type": "STRING", "allowNull": false, "productId": { "type": "INTEGER", "allowNull": true, "description": { "type": "TEXT", "allowNull": true, "association": { [{"as": "books", "ty...	1

id	name	code	metadata	user_id
1	Product	const { DataTypes } = require('sequelize'); const sequelize = require('../config/db'); const Product = sequelize.define('Product', { productId: { type: DataTypes.INTEGER }, title: { type: DataTypes.STRING, allowNull: false }, ...	{ "attributes": { "price": { "type": "FLOAT", "allowNull": false, "title": { "type": "STRING", "allowNull": false, "productId": { "type": "INTEGER", "allowNull": true, "description": { "type": "TEXT", "allowNull": true, "association": { [{"as": "books", "ty...	1
2	Book	const { DataTypes } = require('sequelize'); const sequelize = require('../config/db'); const Book = sequelize.define('Book', { title: { type: DataTypes.STRING, allowNull: false }, author: { type: DataTypes.STRING, allo...	{ "attributes": { "Pages": { "type": "INTEGER", "allowNull": true, "title": { "type": "STRING", "allowNull": false, "author": { "type": "STRING", "allowNull": false, "publishedYear": { "type": "INTEGER", "allowNull": true, "association": { [{"as": "reverse_books", "type": "belongsToMany", "target": "Product", "foreignKey": "productId"}] } } } } } } }	1

4. Delete the another relation between Product & book→ many-to-many

URL: **POST** <http://localhost:3000/api/relationship/delete>

Postman:



Database → relation deleted successfully from both side

Result Grid				
Filter Rows:				
Edit: Export/Import: Wrap Cell Content: F1				
id	name	code	metadata	user_id
2	Book	const { DataTypes } = require('sequelize'); const sequelize = require('../config/db'); const Book = sequelize.define('Book', { title: { type: DataTypes.STRING, allowNull: false }, author: { type: DataTypes.STRING, allo...	{attributes: {Pages: {type: 'INTEGER', 'allowNull': true}, 'title': {type: 'STRING', 'allowNull': false}, 'author': {type: 'STRING', 'allowNull': false}, 'publishedYear': {type: 'INTEGER', 'allowNull': true}}, 'association': []}	1
1	Product	const { DataTypes } = require('sequelize'); const sequelize = require('../config/db'); const Product = sequelize.define('Product', { productId: { type: DataTypes.INTEGER }, title: { type: DataTypes.STRING, allowNull: false }, ...	{attributes: {price: {type: 'FLOAT', 'allowNull': false}, 'title': {type: 'STRING', 'allowNull': false}, 'productId': {type: 'INTEGER', 'allowNull': true}, 'description': {type: 'TEXT', 'allowNull': true}}, 'association': []}	1
NULL	NULL	NULL	NULL	NULL