

API Creation- Mathew Jones

by Mathew Jones

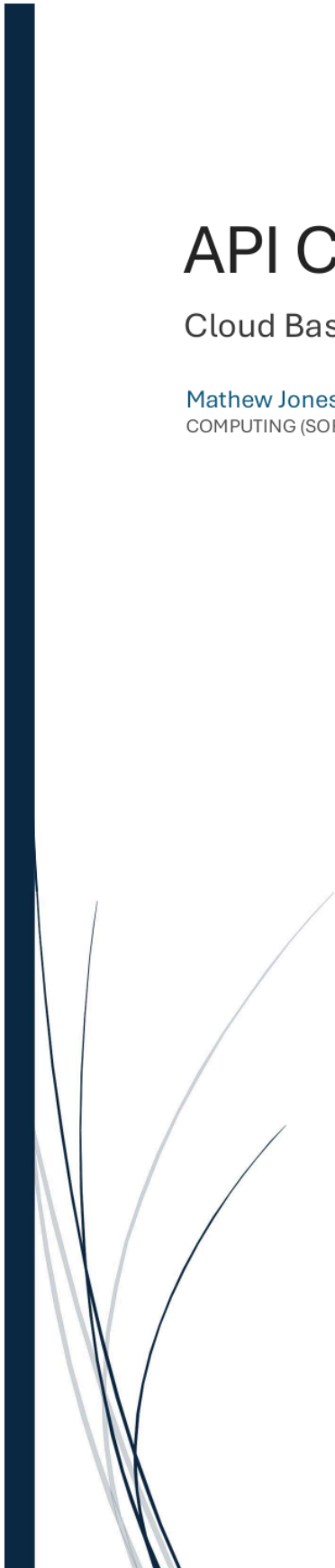
Submission date: 15-Dec-2025 02:15PM (UTC+0000)

Submission ID: 271104942

File name: 50104_Mathew_Jones_API_Creation-_Mathew_Jones_708807_1672202758.docx (492.73K)

Word count: 1577

Character count: 8587



API Creation

Cloud Based Web Services

Mathew Jones (2317425)

COMPUTING (SOFTWARE ENGINEERING)

Contents

Introduction	2
REST APIs	2
Advantages of REST APIs	2
Disadvantages of Rest APIs	2
REST APIs use cases	2
SOAP	3
Advantages of SOAP	3
Disadvantages of SOAP	3
SOAP Use Cases	3
GraphQL	4
Advantages of GraphQL	4
Disadvantages of GraphQL.....	4
GraphQL Use Cases	4
Demonstrate basic operations	5
Implementing a PATCH operation	7
Workflow process	8
Software Testing	10
References	11

Introduction

Modern software applications are often required to communicate with each other and the current method they communicate are APIs or application programming interfaces. They define how data can be requested and exchange between each system. There are many APIs but one of the most common APIs is REST API which will be discussed about in the paper as well as SOAP and GraphQL which are also APIs. They 3 APIs will be compared looking at the advantages, disadvantages and their best use cases. Additionally, there will be evidence in regard to creating and an API, using operations such as GET, POST, PATCH and DELETE.

REST APIs

REST stands for representational state transfer and is an architectural style which is used to connect distributed hypermedia systems. They allow applications to access resources inside of another application, service or database [1]. REST APIs use HTTP methods such as GET, POST, PUT, PATCH and Delete to perform actions such as creating, retrieving, updating or deleting information, these four are commonly known as CRUD.

Advantages of REST APIs

REST APIs have a highly compatible with other applications as it uses the same language, code and architecture. This also makes it very flexible as it can adapt to most applications. They are able to handle lots of information and still operate fine. They use tokens for authentication, which is difficult to breach as they are unique, additionally you are able to improve security by using another authentication method. Finally, APIs have easy and a lot simpler than other APIs [2].

Disadvantages of Rest APIs

The disadvantages of REST APIs are they can be more complex and familiarity with other web architecture is advised. You must be connected to the web to execute operations; this can only be an issue has its performance can be negatively affected by poor internet speeds [2].

REST APIs use cases

The use cases for REST APIs are:

- Web services
- Mobile applications
- IOT devices
- E-Commerce platforms [3]

SOAP

SOAP also known as simple object access protocol is a protocol used to exchange information between different systems. It used XML to format data and commonly follows strict rules which are defined in a contract called WSDL or Web services description language. It is used often in enterprise systems as they require higher level of security and reliability [4].

Advantages of SOAP

SOAP uses encryption and authentication which increases security greatly and makes it suitable for systems which handle sensitive data. There is a strict structure so the server and client should know how messages are to be formatted. Additionally, SOAP supports transaction handling and built-in error handling, this useful for when systems become more complex [5].

Disadvantages of SOAP

The disadvantages of SOAP are it's more complex, so development is slower and performance is impacted as there is more overhead. Due to these reasons SOAP is also harder implement, test and are less flexible. SOAP is not as popular than other APIs due to these reasons [5].

SOAP Use Cases

The use cases for SOAP APIs are:

- Banking systems
- Financial transactions
- Booking flights
- Companies require strict validation [6]

GraphQL

GraphQL was developed by Facebook and is a query-based API. It typically uses a single endpoint, and the client specifies what data it requires, and the server returns only the specified data [7].

Advantages of GraphQL

GraphQL is able to specify the exact data you require which avoids a common issue of over-fetching and under-fetching which leads to an efficient data fetching system. When data requires accessing or analysing a special language called declarative query language can be used to clearly explain what information is required without worrying about technical details of how to retrieve it. These complex data requirements can be expressed easier. The Schema used is easy to understand for developers and outlines the exact structure and data types for servers and clients. This assists in removing confusion and ensures communications are smooth between different areas of the application. Additionally, as stated above GraphQL uses a single endpoint which makes the whole process a lot simpler [8].

Disadvantages of GraphQL

When initially using GraphQL for the first time it can be quite a learning curve as setting up the schema is more complex. There is also a potential for over-fetching which will negatively impact the performance, care is to be taken to prevent this. Security can also be an issue; to prevent this rate limiting and query validations are important security measures to protect against attacks [8].

GraphQL Use Cases

The use cases for GraphQL are:

- Social media platforms
- Mobile and web apps
- Real-time dashboards and data aggregation
- API aggregation and versioning
- Data caching [9]

Demonstrate basic operations

Below demonstrates the POST operation. Entering the game details into title, price, currency, platform and then entering execute will input the data into then API

POST

/v1/games Create Game

Parameters

No parameters

Request body required

application/json

Edit Value | Schema

```
{  "title": "Skyrim",  "price": 29,  "currency": "GBP",  "platforms": [    "Steam"  ]}
```

Execute

Clear

The image below demonstrates the GET operation. Clicking the execute the button will show all games inputted into the API

GET

/v1/games List Games

Parameters

No parameters

Execute

Clear

Responses

Curl

curl -X 'GET' \ "http://localhost:8080/v1/games" \ -H 'accept: application/json'

Request URL

http://localhost:8080/v1/games

Server response

Code | Details

200

Response body

```
{  "title": "Skyrim",  "price": 29,  "currency": "GBP",  "platforms": [    "Steam"  ],  "id": "113997c2-b85d-4b67-ab9b-bf95d4e11554"}
```

Download

To remove a game from the API, the Delete operation will need to be used. Below is evidence of the DELETE operation being used and then checking with the GET operation that the game has actually been deleted from the API.

DELETE

/v1/games/{game_id} Delete Game

Parameters

Cancel

Name	Description
game_id	
string	
(path)	

113997c2-b85d-4b67-ab9b-bf95d4e11554

ExecuteClear

Responses

Curl

```
curl -X 'DELETE' \
  'http://localhost:8000/v1/games/113997c2-b85d-4b67-ab9b-bf95d4e11554' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:8000/v1/games/113997c2-b85d-4b67-ab9b-bf95d4e11554
```

Server response

Code	Details
204	Response headers

GET

/v1/games List Games

Parameters

Cancel

No parameters

ExecuteClear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8000/v1/games' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:8000/v1/games
```

Server response

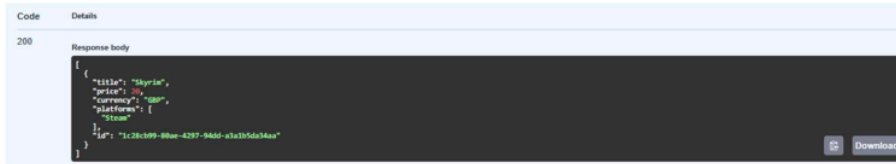
Code	Details
200	Response body

[]

Download

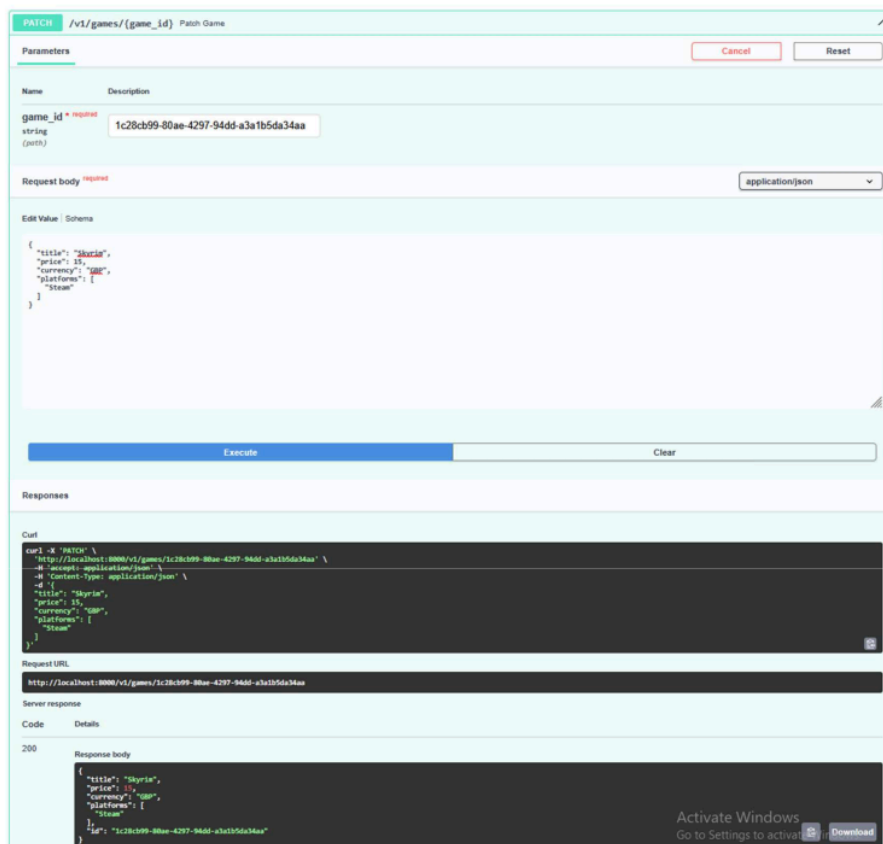
Implementing a PATCH operation

To demonstrate the PUT operation another Skyrim game was inputted into the API database.



```
200
{
  "title": "Skyrim",
  "price": 20,
  "currency": "GBP",
  "platforms": [
    "Steam"
  ],
  "id": "1c28cb99-80ae-4297-94d6-a3a1b5da34aa"
}
```

Using the PATCH operation the price of the game was changed from 20 pounds to 15 pounds.



PATCH /v1/games/{game_id} Patch Game

Parameters

game_id * required
string (path)
1c28cb99-80ae-4297-94d6-a3a1b5da34aa

Request body required
application/json

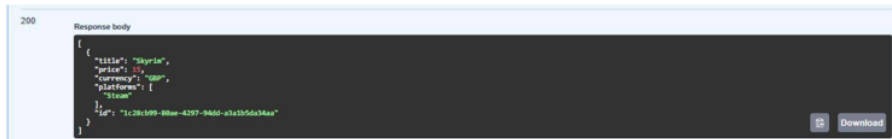
Execute

Responses

200

```
{
  "title": "Skyrim",
  "price": 15,
  "currency": "GBP",
  "platforms": [
    "Steam"
  ],
  "id": "1c28cb99-80ae-4297-94d6-a3a1b5da34aa"
}
```

Below is evidence from the Get operation that it has successfully changed to 15 pounds.



```
200
{
  "title": "Skyrim",
  "price": 15,
  "currency": "GBP",
  "platforms": [
    "Steam"
  ],
  "id": "1c28cb99-80ae-4297-94d6-a3a1b5da34aa"
}
```

Workflow process

Below is an order being create by using the ID of a game and a POST operation.

The screenshot shows a REST client interface for a POST request to `/v1/orders`. The request body is a JSON object containing a `game_id` and a `game_name`. The response is a 201 status code with a JSON body containing the created order details.

```
POST /v1/orders Create Order
```

Parameters: No parameters

Request body *required*: application/json

Edit Value | Schema

```
{  "game_id": [    "ab7be05a-b243-4f55-b4d1-d0b9f2cbb74b"  ]}
```

Execute Clear

Responses

Curl

```
curl -X 'POST' \  "http://localhost:8080/v1/orders" \  -H 'accept: application/json' \  -H 'Content-Type: application/json' \  -d '{    "game_id": [      "ab7be05a-b243-4f55-b4d1-d0b9f2cbb74b"    ]  }'
```

Request URL: `http://localhost:8080/v1/orders`

Server response

Code	Details
201	Response body <pre>{ "id": "ab7be05a-b243-4f55-b4d1-d0b9f2cbb74b", "game_id": ["ab7be05a-b243-4f55-b4d1-d0b9f2cbb74b"], "status": "created"}</pre>

Download

This Get operation will check the status of the order, the order has been created but has not yet been delivered.

The screenshot shows a REST client interface for a GET request to `/v1/orders/{order_id}`. The request path includes the order ID. The response is a 200 status code with a JSON body containing the order details.

```
GET /v1/orders/{order_id} Get Order
```

Parameters

Name	Description
order_id <i>required</i>	

string (path)

ab7be05a-b243-4f55-b4d1-d0b9f2cbb74b

Execute Clear

Responses

Curl

```
curl -X 'GET' \  "http://localhost:8080/v1/orders/ab7be05a-b243-4f55-b4d1-d0b9f2cbb74b" \  -H 'accept: application/json'
```

Request URL: `http://localhost:8080/v1/orders/ab7be05a-b243-4f55-b4d1-d0b9f2cbb74b`

Server response

Code	Details
200	Response body <pre>{ "id": "ab7be05a-b243-4f55-b4d1-d0b9f2cbb74b", "game_id": ["ab7be05a-b243-4f55-b4d1-d0b9f2cbb74b"], "status": "created"}</pre>

Download

This PUT operation will update the Order ID status to delivered.

The screenshot shows a REST client interface for a PUT operation. The URL is `/v1/orders/{order_id}/status` with the description "Deliver Order". The parameter `order_id` is set to `ab7be05a-b243-4f55-b4d1-d0b92cbb74b`. The request body is a JSON object: `{ "status": "delivered" }`. The response is a 200 status code with a JSON body: `{ "id": "ab7be05a-b243-4f55-b4d1-d0b92cbb74b", "game_id": { "id": "6b229008-4877-45b8-9989-e4c12e7b8051", "status": "delivered" } }`.

```
PUT /v1/orders/{order_id}/status Deliver Order

Parameters
Name      Description
order_id  * required
string    ab7be05a-b243-4f55-b4d1-d0b92cbb74b
(path)

Request body  * required
application/json

Edit Value | Schema
{
  "status": "delivered"
}

Execute Clear

Responses

Curl
curl -X PUT \
  "http://localhost:8080/v1/orders/ab7be05a-b243-4f55-b4d1-d0b92cbb74b/status" \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "status": "delivered"
  }'

Request URL
http://localhost:8080/v1/orders/ab7be05a-b243-4f55-b4d1-d0b92cbb74b/status

Server response
Code  Details
200   Response body
{
  "id": "ab7be05a-b243-4f55-b4d1-d0b92cbb74b",
  "game_id": {
    "id": "6b229008-4877-45b8-9989-e4c12e7b8051",
    "status": "delivered"
  }
}
```

Below is just evidence of the PUT operation successfully updating the status of the item.

The screenshot shows a REST client interface for a GET operation. The URL is `/v1/orders/{order_id}` with the description "Get Order". The parameter `order_id` is set to `ab7be05a-b243-4f55-b4d1-d0b92cbb74b`. The response is a 200 status code with a JSON body: `{ "id": "ab7be05a-b243-4f55-b4d1-d0b92cbb74b", "game_id": { "id": "6b229008-4877-45b8-9989-e4c12e7b8051", "status": "delivered" } }`.

```
GET /v1/orders/{order_id} Get Order

Parameters
Name      Description
order_id  * required
string    ab7be05a-b243-4f55-b4d1-d0b92cbb74b
(path)

Execute Clear

Responses

Curl
curl -X GET \
  "http://localhost:8080/v1/orders/ab7be05a-b243-4f55-b4d1-d0b92cbb74b" \
  -H 'accept: application/json'

Request URL
http://localhost:8080/v1/orders/ab7be05a-b243-4f55-b4d1-d0b92cbb74b

Server response
Code  Details
200   Response body
{
  "id": "ab7be05a-b243-4f55-b4d1-d0b92cbb74b",
  "game_id": {
    "id": "6b229008-4877-45b8-9989-e4c12e7b8051",
    "status": "delivered"
  }
}
```

Software Testing

Test Title:	Create game
Test Author:	Mathew Jones
Test Description:	This test verifies that a new game can be created using the POST operation. The user submits valid game data including the title, price, currency and platform the game is on.
Expected Outcome:	The API returns HTTP 201 created and responds with the newly created game including a unique ID

Test Title:	List Games
Test Author:	Mathew Jones
Test Description:	This test verifies that all stored game can be retrieved using the GET operation.
Expected Outcome:	The API returns HTTP 200 ok and a list containing all games currently stored in the system.

Test Title:	Update game using PATCH
Test Author:	Mathew Jones
Test Description:	This test verifies that a game can be partially updated using the PATCH operation. Only the fields provided in the request are modified.
Expected Outcome:	The API returns HTTP 200 ok and only the specific fields are updated while all other fields remain unchanged.

Test Title:	Create order
Test Author:	Mathew Jones
Test Description:	This test verifies the first stage of the workflow where an order is created containing a list of game IDs using the POST operation
Expected Outcome:	The API returns HTTP 201 created, and the order is stored with its stat set to "Created".

Test Title:	Deliver order
Test Author:	Mathew Jones
Test Description:	This test verifies the second stage of the workflow where an existing order is marked as delivered using the PUT operation.
Expected Outcome:	The API returns HTTP 200 ok and the order status change from "created" to "delivered".

Test Title:	Replace game using PUT
Test Author:	Mathew Jones
Test Description:	This test verifies that an existing game can be fully replaced using PUT endpoint.
Expected Outcome:	The API returns HTTP 200 ok and the game is updated with the new values provided.

Test Title:	Delete Game
Test Author:	Mathew Jones
Test Description:	This test verifies that an existing game can be deleted using the DELETE endpoint.
Expected Outcome:	The API returns HTTP 204 "no content" and the game is removed from the system.

References

- [1] IBM Team. "What Is a REST API (RESTful API)? | IBM." IBM. Accessed: Dec 12 2025. [Online]. Available: <https://www.ibm.com/think/topics/rest-apis>
- [2] PowerMBA Team. "What Is a REST API? Definition, Benefits & Examples." ThePowerMBA. Accessed: Dec 15 2025. [Online]. Available: <https://www.thepowermba.com/en/blog/rest-api-what-it-is>
- [3] BrowserStack Team. "What is a REST API? Examples, Use Cases, and Best Practices." BrowserStack. Accessed: Dec 13 2025. [Online]. Available: <https://browserstack.wpengine.com/guide/rest-api/>
- [4] Lane Kin. "What Is a SOAP API?" Postman. Accessed: Dec 15 2025. [Online]. Available: <https://blog.postman.com/soap-api-definition/>
- [5] AWS Amazon Team. "What is Cloud Computing? Cloud Computing Concepts Hub Application Integration What's the Difference Between SOAP and REST?" AWS Amazon. Accessed: Dec 14 2025. [Online]. Available: <https://aws.amazon.com/compare/the-difference-between-soap-rest/>
- [6] B. Pedro. "Nordic APIs | Cases When Using SOAP Makes Sense" *Nordic APIs*. (-03-022015). Accessed: Dec 14, 2025. [Blog]. Available at: <https://nordicapis.com/common-cases-when-using-soap-makes-sense/>
- [7] Hygraph Team. "What Is GraphQL and How It Works - GraphQL Academy." Hygraph. Accessed: Dec 14 2025. [Online]. Available: <https://hygraph.com/learn/graphql>
- [8] GeeksforGeeks Team. "Advantage & Disadvantage of GraphQL." GeeksforGeeks. Accessed: Dec 14 2025. [Online]. Available: <https://www.geeksforgeeks.org/graphql/advantage-disadvantage-of-graphql/>
- [9] KongHQ Team. "What is GraphQL? Use Cases and Examples." KongHQ. Accessed: Dec 15 2025. [Online]. Available: <https://konghq.com/blog/learning-center/graphql>

FINAL GRADE

GENERAL COMMENTS

30/100

Generic Assessment Criteria	Marks available	Marks awarded
1. Engagement with Literature Skills .	Assignment 1: 5	
A reasonable number of sources. Some academic and credible sources amongst the websites would have been desirable.		2.5
2. Knowledge and Understanding Skills	Assignment 1: 10	
Investigation, use-cases and advantages / disadvantages. Reasonable knowledge demonstrated of REST, SOAP and GraphQL. A more detailed discussion of the workings would have been beneficial.		6

Comparisons made between the three would have been beneficial, more detailed discussion around the use cases would have been advantageous, although some good use cases displayed in point form.		
1. Cognitive and Intellectual Skills	Assignment 1: 30	
Design and documentation Test documentation and testing A reasonable demonstration in the report of the CRUD operations. Use of test tables is ok and shows some evidence of testing, although further boxes for PASS or FAIL and actual outcome would have been a desirable addition.		20
4. Practical Application Skills	Assignment 1: 50	
AJAX implementation REST API No code files or folder provided, only a .docx file with python code. Javascript Express API was required here.		0

5. Transferable Skills for Life and Professional Practice	Assignment 1: 5	
Presentation Reasonable presentation, would have benefited conclusion. Captioning the figures and tables would have been desirable. Attention to detail on the grammar may have given a better flow to the report.		1.5

PAGE 1
PAGE 2
PAGE 3
PAGE 4
PAGE 5
PAGE 6
PAGE 7
PAGE 8
PAGE 9
PAGE 10
PAGE 11
PAGE 12