Charter

Identify capabilities and areas of potential instability of the “rest api todo list manager”.
Identify documented and undocumented “rest api todo list manager” capabilities.
For each capability create a script or small program to demonstrate the capability.
Exercise each capability identified with data typical to the intended use of the application.

Build

```
java -jar runTodoManagerRestAPI-1.5.5.jar
```

Area

Capabilities and areas of potential instability (documented and undocumented) as well as tests
using data typical to the intended use of the application.

Environment

Windows 10, 1920x1080 (Joey)
MacOS 13.4.1, 3024 × 1964 (Ke)

Start

2:53pm, Monday October 9th 2023

Testers

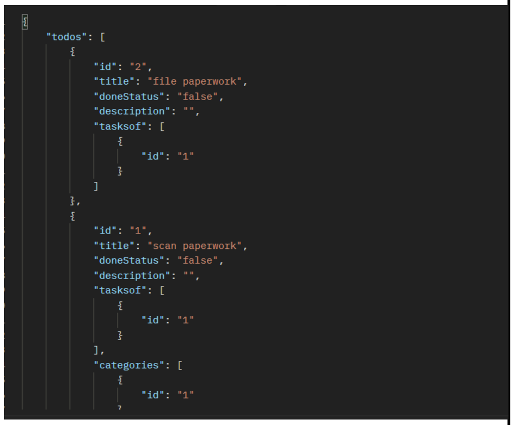
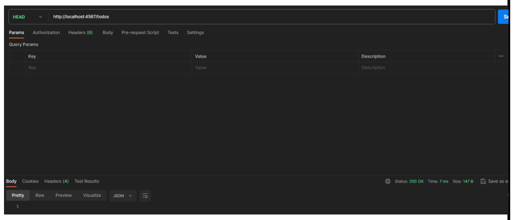
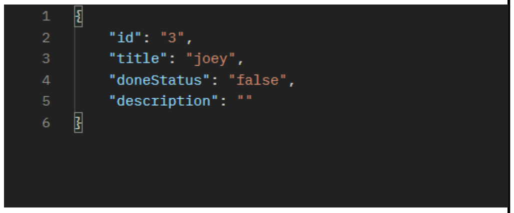
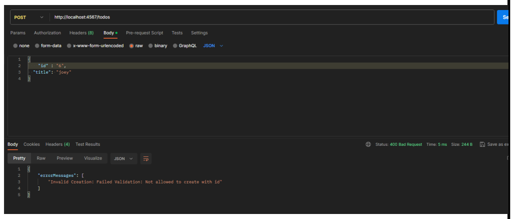
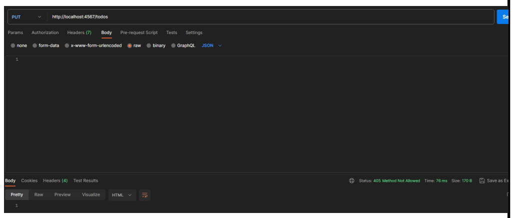
Ke Yan	260988148	ke.yan@mail.mcgill.ca
Joey Liu	260969992	lucas.liuliongwah@mail.mcgill.ca

Session Breakdown

Total length 45 minutes

Test Notes

Session 1

Time	Tester	Request	Type	Comment	Screenshot
2:55	Joey	http://localhost:4567/todos	GET	All instances of the todos object are retrieved. (**Works as expected**)	
2:57			HEAD	Receive the response OK and code 200, no output in body. (**Works as expected**)	
3:00			POST	A new todo object with id 3 was created. (**Works as expected**)	
3:01			POST(todo id : 6)	Cannot create a new todo object if I pass the Id through the body message. (**Does not works as expected**)	
3:02			PUT (undocumented)	Received code 405 and “Method Not Allowed” as message. (**Works as expected**)	

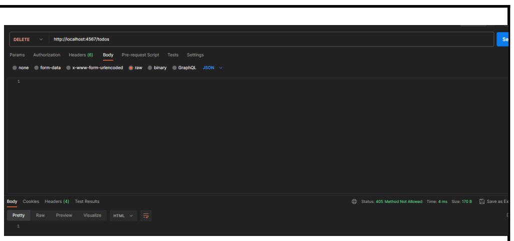
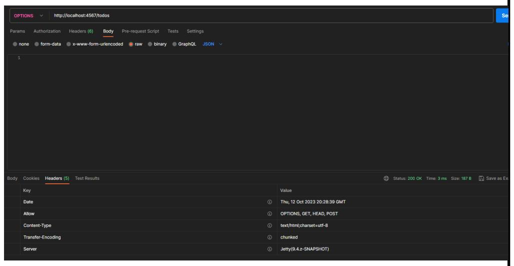
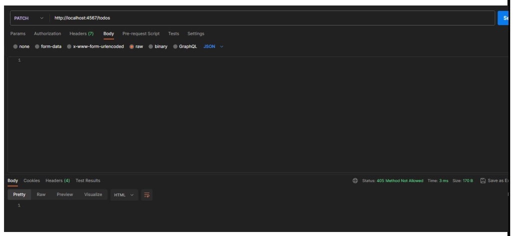
```
"todos": [
    {
        "id": "2",
        "title": "file paperwork",
        "doneStatus": "false",
        "description": "",
        "tasksof": [
            {
                "id": "1"
            }
        ]
    },
    {
        "id": "1",
        "title": "scan paperwork",
        "doneStatus": "false",
        "description": "",
        "tasksof": [
            {
                "id": "1"
            }
        ],
        "categories": [
            {
                "id": "1"
```

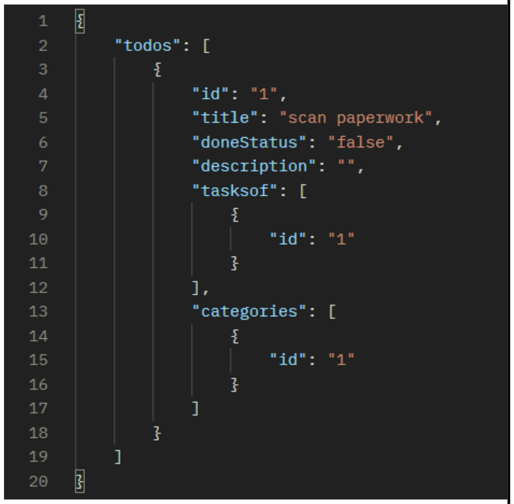
```
HEAD    http://localhost:4567/todos
Params  Authorization  Headers (6)  Body  Pre-request Script  Tests  Settings
Query Params
Key    Value    Description
Key    Value    Description

Body  Cookies  Headers (4)  Test Results        Status: 200 OK  Time: 7 ms  Size: 147 B  Save as
Pretty  Raw  Preview  Visualize  JSON
```

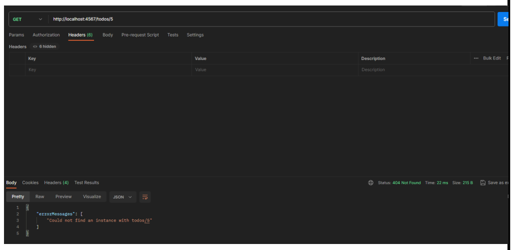
```
{
    "id": "3",
    "title": "joey",
    "doneStatus": "false",
    "description": ""
}
```

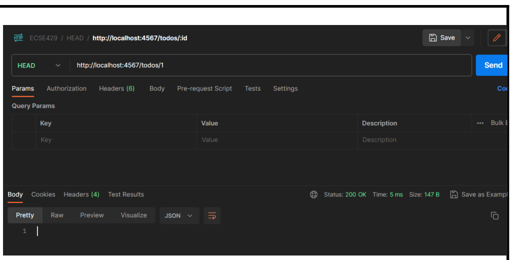
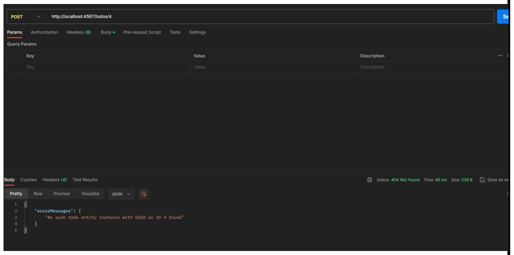
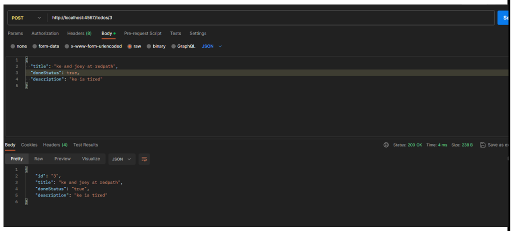
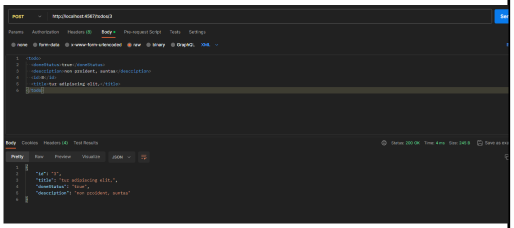
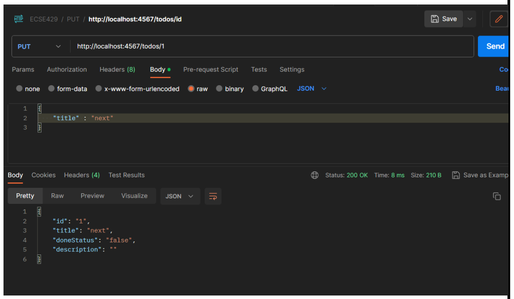
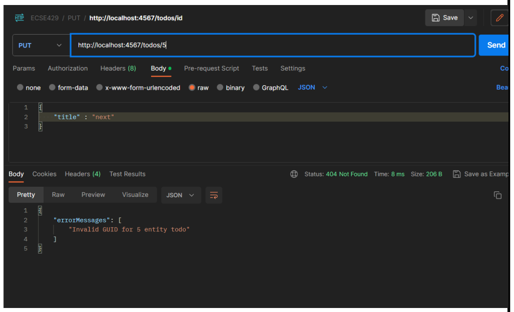
```
POST    http://localhost:4567/todos
Params  Authorization  Headers (8)  Body  Pre-request Script  Tests  Settings
none  form-data  x-www-form-urlencoded  raw  binary  GraphQL  JSON
{
    "id" : "6",
  "title": "joey"
}

Body  Cookies  Headers (4)  Test Results        Status: 400 Bad Request  Time: 5 ms  Size: 244 B  Save as ex
Pretty  Raw  Preview  Visualize  JSON
{
    "errorMessages": [
        "Invalid Creation: Failed Validation: Not allowed to create with id"
    ]
}
```

```
PUT    http://localhost:4567/todos
Params  Authorization  Headers (7)  Body  Pre-request Script  Tests  Settings
none  form-data  x-www-form-urlencoded  raw  binary  GraphQL  JSON

Body  Cookies  Headers (4)  Test Results        Status: 405 Method Not Allowed  Time: 76 ms  Size: 170 B  Save as Ex
Pretty  Raw  Preview  Visualize  HTML
```

3:02			DELETE (undocumented)	Received code 405 and “Method Not Allowed” as message. (**Works as expected**)	 DELETE http://localhost:4567/todos Params Authorization Headers (6) Body Pre-request Script Tests Settings none form-data x-www-form-urlencoded raw binary GraphQL JSON Body Cookies Headers (4) Test Results Status: 405 Method Not Allowed Time: 4 ms Size: 170 B Pretty Raw Preview Visualize HTML
3:05			OPTIONS (undocumented)	Received code 200 and “OK” as message. (**Works as expected**) We can see in the Allow header, all the endpoints that can be performed	 OPTIONS http://localhost:4567/todos Params Authorization Headers (6) Body Pre-request Script Tests Settings none form-data x-www-form-urlencoded raw binary GraphQL JSON Body Cookies Headers (5) Test Results Status: 200 OK Time: 3 ms Size: 187 B Key / Value Date: Thu, 12 Oct 2023 20:28:39 GMT Allow: OPTIONS, GET, HEAD, POST Content-Type: text/html;charset=utf-8 Transfer-Encoding: chunked Server: Jetty(9.4.z-SNAPSHOT)
3:08			PATCH (undocumented)	Received code 405 and “Method Not Allowed” as message. (**Works as expected)**	 PATCH http://localhost:4567/todos Params Authorization Headers (7) Body Pre-request Script Tests Settings none form-data x-www-form-urlencoded raw binary GraphQL JSON Body Cookies Headers (4) Test Results Status: 405 Method Not Allowed Time: 3 ms Size: 170 B Pretty Raw Preview Visualize HTML

3:10		http://localhost:4567/todos/id	GET
			(Id: 1)
				Returns object todo with Id 1.
				(**Works as expected**)
					
					`{`
					`"todos": [`
					`{`
					`"id": "1",`
					`"title": "scan paperwork",`
					`"doneStatus": "false",`
					`"description": "",`
					`"tasksof": [`
					`{`
					`"id": "1"`
					`}`
					`],`
					`"categories": [`
					`{`
					`"id": "1"`
					`}`
					`]`
					`}`
					`]`
					`}`

3:11			GET
			(with invalid Id)
				Received code 404 and “Not Found” as message. Error Message: Could not find an instance with todos/5
				(**Works as expected**)
					
					GET http://localhost:4567/todos/5
					Params Authorization Headers (6) Body Pre-request Script Tests Settings
					Headers 6 hidden
					Key Value Description Bulk Edit
					Body Cookies Headers (4) Test Results
					Status: 404 Not Found Time: 22 ms Size: 215 B
					Pretty Raw Preview Visualize JSON
					`{`
					`"errorMessages": [`
					`"Could not find an instance with todos/5"`
					`]`
					`}`

3:13			HEAD	Received the response OK and code 200, no output in body. (**Works as expected**)	
3:16			POST (Id not in memory)	Received code 404 and “Not Found” as message. Error Message: No such todo entity instance with GUID or ID 4 found (**Works as expected**)	
3:17			POST (Id in memory)	Received the response OK and code 200, new todo object is the response that we got. (**Works as expected**)	
3:18			POST (XML) (Id in memory)	Added id 8 in body message but still amended the new object with id 3. (**Does not works as expected**)	
3:20			PUT	Received the response OK and code 200, new todo object is the response that we got. (**Works as expected**)	
3:22			PUT (Id not in memory)	Received code 404 and “Not Found” as message. Error Message: Invalid GUID for 5 entity todo. (**Works as expected**)	

ECSE429 / HEAD / http://localhost:4567/todos/:id

HEAD http://localhost:4567/todos/1

Query Params: Key | Value | Description

Status: 200 OK Time: 5 ms Size: 147 B

POST http://localhost:4567/todos/4

Status: 404 Not Found Time: 85 ms Size: 228 B

```
{
    "errorMessages": [
        "No such todo entity instance with GUID or ID 4 found"
    ]
}
```

POST http://localhost:4567/todos/3

Body: raw JSON

```
{
  "title": "ke and joey at redpath",
  "doneStatus": true,
  "description": "ke is tired"
}
```

Status: 200 OK Time: 4 ms Size: 238 B

```
{
    "id": "3",
    "title": "ke and joey at redpath",
    "doneStatus": "true",
    "description": "ke is tired"
}
```

POST http://localhost:4567/todos/3

Body: raw XML

```
<todo>
  <doneStatus>true</doneStatus>
  <description>non proident, suntaa</description>
  <id>8</id>
  <title>tur adipiscing elit,</title>
</todo>
```

Status: 200 OK Time: 4 ms Size: 245 B

```
{
    "id": "3",
    "title": "tur adipiscing elit,",
    "doneStatus": "true",
    "description": "non proident, suntaa"
}
```

ECSE429 / PUT / http://localhost:4567/todos/id

PUT http://localhost:4567/todos/1

Body: raw JSON

```
{
    "title" : "next"
}
```

Status: 200 OK Time: 8 ms Size: 210 B

```
{
    "id": "1",
    "title": "next",
    "doneStatus": "false",
    "description": ""
}
```

ECSE429 / PUT / http://localhost:4567/todos/id

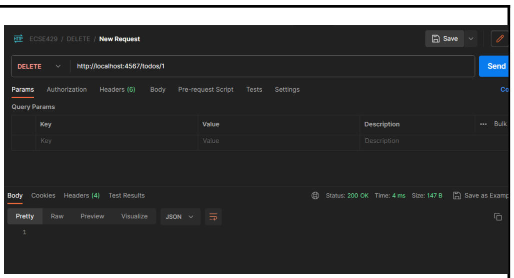
PUT http://localhost:4567/todos/5

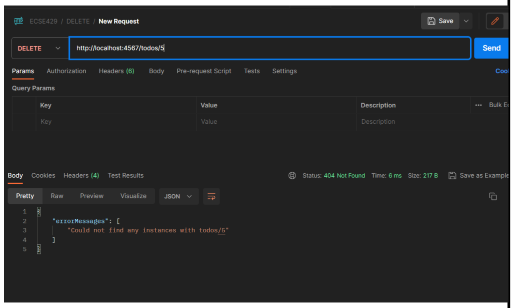
Body: raw JSON

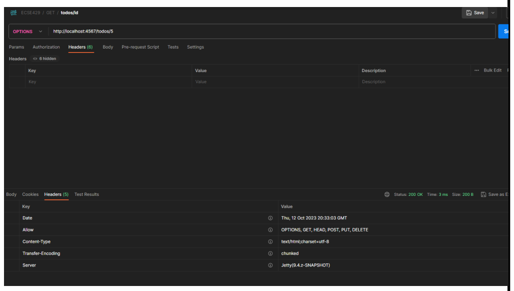
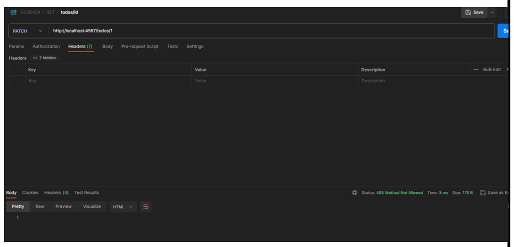
```
{
    "title" : "next"
}
```

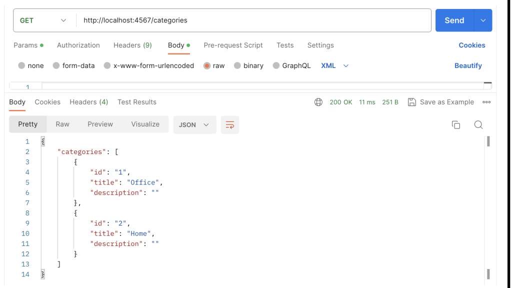
Status: 404 Not Found Time: 8 ms Size: 206 B

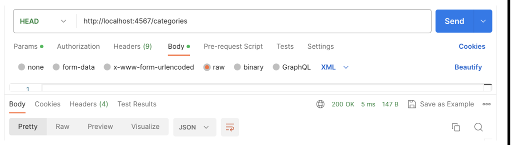
```
{
    "errorMessages": [
        "Invalid GUID for 5 entity todo"
    ]
}
```

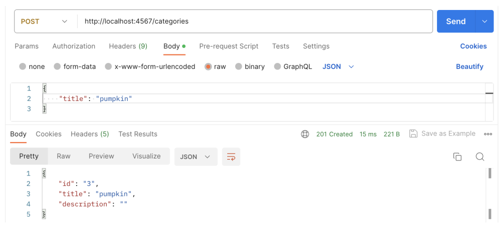
3:23			DELETE (Id in memory)	Received the response OK and code 200, new todo object is the response that we got. (**Works as expected**)	 ECSE429 / DELETE / New Request DELETE http://localhost:4567/todos/1 Params Authorization Headers (6) Body Pre-request Script Tests Settings Query Params: Key / Value / Description Body Cookies Headers (4) Test Results — Status: 200 OK Time: 4 ms Size: 147 B Pretty Raw Preview Visualize JSON

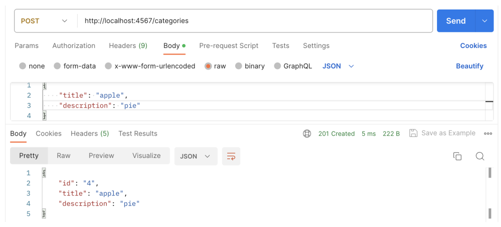
3:24			DELETE
			(Id not in memory)
				Received code 404 and "Not Found" as message. Error Message: Could not find an instance with todos/5
				(**Works as expected**)
					
					ECSE429 / DELETE / New Request
					DELETE http://localhost:4567/todos/5
					Params Authorization Headers (6) Body Pre-request Script Tests Settings
					Query Params: Key / Value / Description
					Body Cookies Headers (4) Test Results — Status: 404 Not Found Time: 6 ms Size: 217 B
					Pretty Raw Preview Visualize JSON
					`{`
					`    "errorMessages": [`
					`        "Could not find any instances with todos/5"`
					`    ]`
					`}`

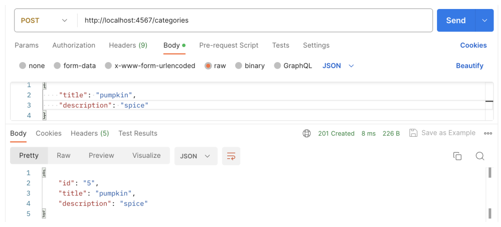
3:25			OPTIONS (Undocumented)	Received code 200 and "OK" as message. (**Works as expected**) We can see in the Allow header, all the endpoints that can be performed	 ECSE429 / GET / todos/id OPTIONS http://localhost:4567/todos/5 Params Authorization Headers (6) Body Pre-request Script Tests Settings Headers (6 hidden): Key / Value / Description Body Cookies Headers (5) Test Results — Status: 200 OK Time: 3 ms Size: 200 B Date: Thu, 12 Oct 2023 20:33:03 GMT Allow: OPTIONS, GET, HEAD, POST, PUT, DELETE Content-Type: text/html;charset=utf-8 Transfer-Encoding: chunked Server: Jetty(9.4.z-SNAPSHOT)
3:26			PATCH (Undocumented)	Received code 404 and "Not Found" as message. Error Message: Could not find an instance with todos/5 (**Works as expected**)	 ECSE429 / GET / todos/id PATCH http://localhost:4567/todos/1 Params Authorization Headers (7) Body Pre-request Script Tests Settings Headers (7 hidden): Key / Value / Description Body Cookies Headers (4) Test Results — Status: 405 Method Not Allowed Time: 3 ms Size: 170 B Pretty Raw Preview Visualize HTML

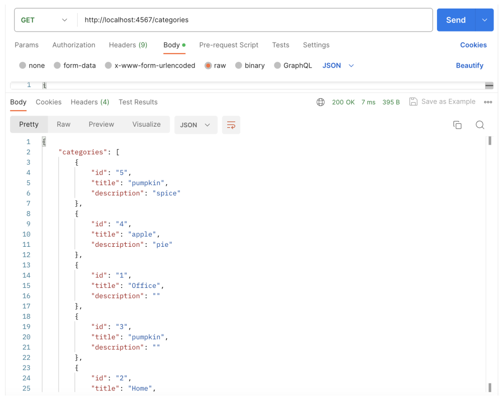
3:28	Ke	http://localhost:4567/categories	GET	Received the existing objects with a 200 code (**Works as expected**)	
					GET http://localhost:4567/categories
					Params Authorization Headers (9) Body Pre-request Script Tests Settings
					none form-data x-www-form-urlencoded raw binary GraphQL XML
					Body Cookies Headers (4) Test Results — 200 OK 11 ms 251 B
					Pretty Raw Preview Visualize JSON
					`{`
					`    "categories": [`
					`        {`
					`            "id": "1",`
					`            "title": "Office",`
					`            "description": ""`
					`        },`
					`        {`
					`            "id": "2",`
					`            "title": "Home",`
					`            "description": ""`
					`        }`
					`    ]`
					`}`

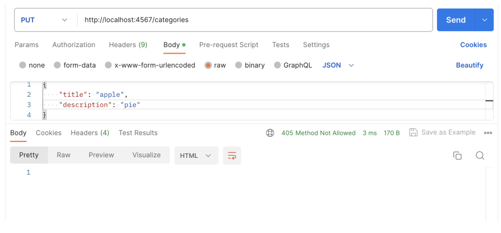
3:29			HEAD	Received the response OK and code 200, no output in body. (**Works as expected**)	 HEAD http://localhost:4567/categories Params Authorization Headers (9) Body Pre-request Script Tests Settings none form-data x-www-form-urlencoded raw binary GraphQL XML Body Cookies Headers (4) Test Results — 200 OK 5 ms 147 B Pretty Raw Preview Visualize JSON

3:31			POST (with title)	A new category object is created. (**Works as expected**)	 POST http://localhost:4567/categories
					{
					"title": "pumpkin"
					}
					201 Created 15 ms 221 B
					{
					"id": "3",
					"title": "pumpkin",
					"description": ""
					}

3:32			POST (with title + description)	A new category object is created. (**Works as expected**)	 POST http://localhost:4567/categories
					{
					"title": "apple",
					"description": "pie"
					}
					201 Created 5 ms 222 B
					{
					"id": "4",
					"title": "apple",
					"description": "pie"
					}

3:34			POST (existing title)	A new category object is created. (**unexpected, system allows multiple categories with the same name**)	 POST http://localhost:4567/categories
					{
					"title": "pumpkin",
					"description": "spice"
					}
					201 Created 8 ms 226 B
					{
					"id": "5",
					"title": "pumpkin",
					"description": "spice"
					}

3:35			GET (after post)	Returns the initial objects + newly created ones. (**Works as expected**)	 GET http://localhost:4567/categories 200 OK 7 ms 395 B { "categories": [{ "id": "5", "title": "pumpkin", "description": "spice" }, { "id": "4", "title": "apple", "description": "pie" }, { "id": "1", "title": "Office", "description": "" }, { "id": "3", "title": "pumpkin", "description": "" }, { "id": "2", "title": "Home",

3:37			PUT	Received code 405 and "Method Not Allowed" as message. (**Works as expected**)	 PUT http://localhost:4567/categories
					{
					"title": "apple",
					"description": "pie"
					}
					405 Method Not Allowed 3 ms 170 B

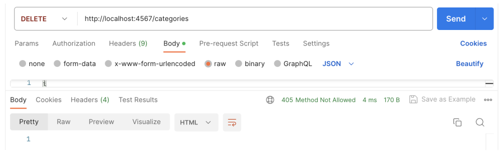
3:38			DELETE	Received code 405 and "Method Not Allowed" as message. (**Works as expected**)	 DELETE http://localhost:4567/categories 405 Method Not Allowed 4 ms 170 B

Table 1: Findings of Session 1

Summary of Session

This session we tested …
/todos

- GET **Works as expected**
- HEAD **Works as expected**
- POST **Works as expected**
    - With id in body **Unexpected behaviour**
- PUT (undocumented) **Works as expected**
- DELETE (undocumented) **Works as expected**
- OPTIONS (undocumented) **Works as expected**
- Patch (undocumented) **Works as expected**

/todos/id

- GET **Works as expected**
- HEAD **Works as expected**
- POST
    - Valid id (JSON) **Works as expected**
    - Invalid id (JSON) **Works as expected**
    - XML **Works as expected**
    - PUT
    - DELETE **Works as expected**
    - OPTIONS (undocumented) **Works as expected**
    - PATCH (undocumented) **Works as expected**

/categories

- GET **Works as expected**
- HEAD **Works as expected**
- POST
    - with title **Works as expected**
    - with title + description **Works as expected**
    - with the same title as an existing object **Unexpected behaviour**
- PUT **Works as expected**
- DELETE **Works as expected**

List of Concerns Identified in Session

- when doing a POST with XML in /todos/id, if we pass in an id in the body, the changes are still amended to the id passed in the parameter. This can be confusing since there are no warnings, and a user could be under the impression that the object with the id t in the body was updated.

Steps to reproduce:

- Create a todo using post, take down id
- Make POST request with the id in the parameter, but a different id in the body

- noticed that multiple categories with the same names are allowed. This could lead to confusion for the user, since they would only be differentiated by id and maybe a description.

Steps to reproduce:

- Create a new category with some name
- Create another category with the same name

List of new Testing Ideas Identified in Session

- Since the session was timeboxed, we were unable to test all possibilities that could lead to errors. We would like to try more cases that could lead to errors, for example deleting a todo that used to exist, but has been deleted previously. That way we can be sure the database is updated accurately. We will add unit tests to todos and categories to ensure that operations do not have side effects. (discussed with Neeraj).