Name	Martin Eskaros
ID	261031885

<u>Charter</u>

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.

<u>Build</u>

java -jar runTodoManagerRestAPI-1.5.5.jar

<u>Area</u>

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

<u>Environment</u>

Mac OS, external display resolution 2560x1440p

<u>Start</u>

10pm, October 7th 2024

<u>Testers</u>

Martin Eskaros 261031885 martin.eskaros@mail.mcgill.ca

<u>Session Breakdown</u>

Length:45 minutes

Using curl to make HTTP requests from command line.

Session 1 – todos commands

Command #1

```
curl --location --request GET 'http://localhost:4567/todos'
```

Command #2

```
curl --location --head 'http://localhost:4567/todos'
```

Command #3

```
curl --location --request POST 'http://localhost:4567/todos' \
--header 'Content-Type: application/json’ \
--data-raw '
{
 "title": "this is a test",
 "doneStatus": false,
 "description": "i love 429"
}'
```

Command #4

```
curl --location --request POST 'http://localhost:4567/todos' \
--header 'Content-Type: application/json ' \
--data-raw '{
 "id": "31",
 "title": "um dolore eu fugiata",
 "doneStatus": "false",
 "description": "ud exercitation ulla"
}'
```

Command #5

```
curl --location --request GET 'http://localhost:4567/todos/14'
```

Command #6

```
curl --location --request GET
'http://localhost:4567/todos?title=velit%20esse%20cillum%20do&description=a%20aliqua.%20Ut%20enim%20ad' \
--data-raw ''
```

Command #7

```
curl --location --request GET
'http://localhost:4567/todos?title=velit%20esse%20cillum%20do&description=a%20aliqua.%20Ut%20enim%20ad'
```

Command #8

```
curl --location --request POST 'http://localhost:4567/todos' \
--header 'Content-Type: application/json' \
--data-raw '{
 "title": "simple",
 "doneStatus": false,
 "description": "example"
}'
```

Command #9

```
curl --location --request GET 'http://localhost:4567/todos/13' \
--header 'Accept: application/xml'--data-raw '<category>
```

Command #10

```
curl --location --request DELETE 'http://localhost:4567/todos/1'
```

Command #11

```
curl --location --request POST 'http://localhost:4567/todos/21' \
--header 'Content-Type: application/json' \
--data-raw '{
    "title": "Amended Todo with ID 21",
    "doneStatus": false,
    "description": "Updated using POST to specific ID"
}'
```

Command #12

```
curl --location --request PUT 'http://localhost:4567/todos/:14' \
--header 'Content-Type: application/json' \
--data-raw ' {

        "title": "this is a test_updating",
        "doneStatus": "false",
        "description": "i love 429"      }'
```

Command #13

```
curl --location --request DELETE 'http://localhost:4567/15' \
--data-raw ''
```

Command #14

```
curl --location --request POST 'http://localhost:4567/todos/3/categories' \
--header 'Content-Type: application/xml' \
--data-raw '<category>
<id>3</id>
</category>'
```

Command #15

```
curl --location --request POST 'http://localhost:4567/todos/3/categories' \
--header 'Content-Type: application/xml' \
--data-raw '<category>
<id>5</id>
</category>'
```

Command #16

```
curl --location --request DELETE 'http://localhost:4567/todos/300' \
--data-raw ''
```

Command #17

```
curl --location --request GET 'http://localhost:4567/todos/2/categories' \
--data-raw ''
```

Command #18

```
curl --location --request PUT 'http://localhost:4567/todos/13' \
--header 'Content-Type: application/xml' \
--data-raw '<category>
<title>hello</title>
<id>3</id>
</category>'
```

Command #19

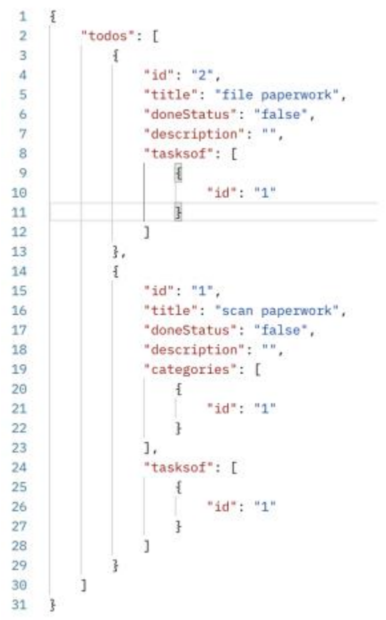
```
curl --location --request GET 'http://localhost:4567/shutdown' \
--data-raw ''
```

See List Of Commands in previous pages

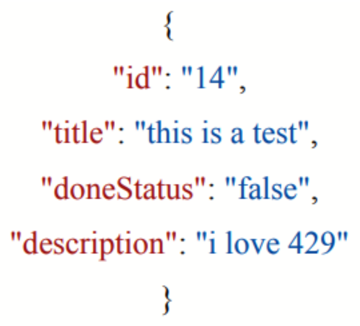
Command	Type	Is Documented?	Expected Behavior
Command #1	GET	DOCUMENTED	Return all to do instances.
Command #2	HEAD	DOCUMENTED	Retrieve headers of the todos.
Command #3	POST	DOCUMENTED	Create a todo without a ID using the field values in the body of the message.
Command #4	POST	UNDOCUMENTED	Not allowed
Command #5	GET	DOCUMENTED	Returns a specific instance of todo using an id.
Command #7	GET	UNDOCUMENTED	Not allowed
Command #8	POST	DOCUMENTED	Create a todo without a ID using the field values in the body of the message.
Command #9	GET	DOCUMENTED	Returns a specific instance of todo using an id.
Command #10	DELETE	DOCUMENTED	Delete a specific instance of todo using an id(1)
Command #11	POST	DOCUMENTED	Amend a specific instances of todo using a id with a body containing the fields to amend
Command #12	PUT	DOCUMENTED	Amend a specific instances of todo using a id with a body containing the fields to amend
Command #13	DELETE	DOCUMENTED	Delete a specific instance of todo using an id(15)
Command #14	POST	UNDOCUMENTED	Not allowed

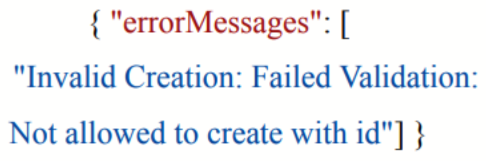
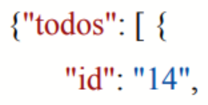
Command #15	POST	DOCUMENTED	Create an instance of a relationship named categories between todo instance :id and the category instance represented by the id in the body of the message
Command #16	DELETE	DOCUMENTED	Delete a specific instance of todo using an id(300)
Command #17	GET	DOCUMENTED	return all the category items related to todo, with given id, by the relationship named categories
Command #18	PUT	DOCUMENTED	amend a specific instances of todo using a id with a body containing the fields to amend
Command #19	GET	DOCUMENTED	Shutdown the API server

Expected == observed	Observed behavior	Side Notes	Output & Screenshots

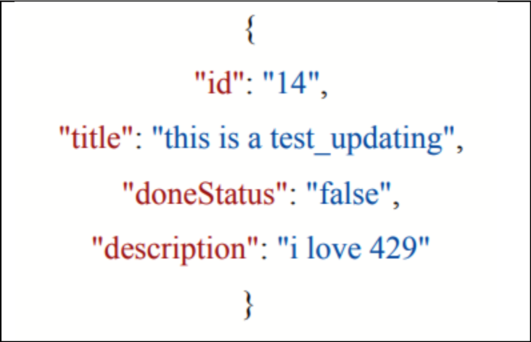
YES	All instances were received.		 {
			"todos": [
			{
			"id": "2",
			"title": "file paperwork",
			"doneStatus": "false",
			"description": "",
			"tasksof": [
			{
			"id": "1"
			}
			]
			},
			{
			"id": "1",
			"title": "scan paperwork",
			"doneStatus": "false",
			"description": "",
			"categories": [
			{
			"id": "1"
			}
			],
			"tasksof": [
			{
			"id": "1"
			}
			]
			}
			]
			}

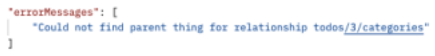
YES	Outputs date, the content-type, transfer enconding and server	Content-type is correctly application/json.	`HTTP/1.1 200 OK`
			`Date:` `Tue, 08 Oct 2024 03:38:12 GMT`
			`Content-Type:` `application/json`
			`Transfer-Encoding:` `chunked`
			`Server:` `Jetty(9.4.z-SNAPSHOT)`

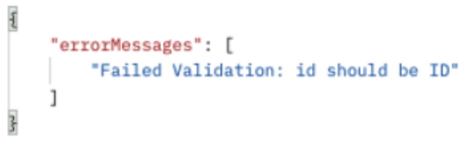
YES	Created a new todo without an Id as documented. An Id of 14 was randomly assigned.		 {
			"id": "14",
			"title": "this is a test",
			"doneStatus": "false",
			"description": "i love 429"
			}

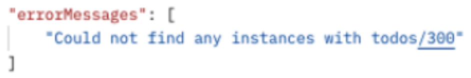
YES	Does not allow the creation of a todo with an ID	I believe this should be allowed.	 { "errorMessages": ["Invalid Creation: Failed Validation: Not allowed to create with id"] }
YES	Retrieved the todo that was created 2 commands ago with id 14.		 {"todos": [{ "id": "14",

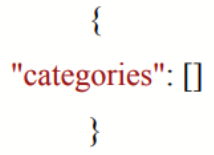
			"title": "this is a test", "doneStatus": "false", "description": "i love 429"}]}
N/A	Output was an empty array of todos.	Not clear how the URL is hashed and can be broken down.	{ "todos": [] }
YES	Created a new to	Will use this later to try and retrieve by URL queury	{ "id": "15", "title": "simple", "doneStatus": "false", "description": "example" }
YES	Returned information for todo with provided ID value(13)		{ "todos": [{ "id": "13", "title": "Office Work Project", "doneStatus": "false", "description": "" }] }
YES	Deleted the todo with id		N/A
No	Got error message.	Does not seem like you can post a todo with an id in the endpoint.	"errorMessages":["No such todo entity instance with GUID or ID 21 found"]

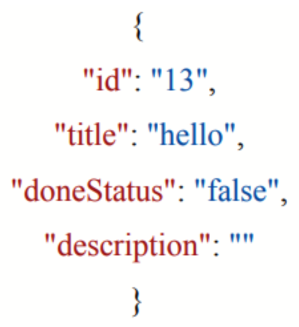
YES	Updated the title of todo instance id 14	Title was updated to : “this is a test_updating”	 {
			"id": "14",
			"title": "this is a test_updating",
			"doneStatus": "false",
			"description": "i love 429"
			}

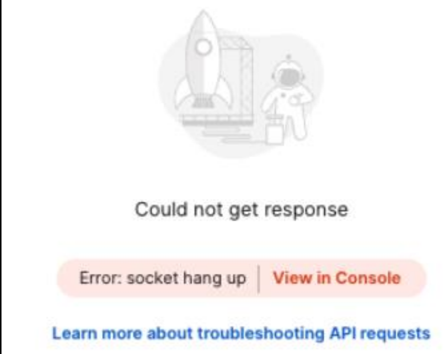
YES	Deleted todo instance with ID 15		N/A
N/A	Got error message.	Was unable to POST using XML input for valid ID value, relationship not identified	 "errorMessages": ["Could not find parent thing for relationship todos/3/categories"]

N/A	Got error message.	Failed validation.	 {
			"errorMessages": [
			"Failed Validation: id should be ID"
			]
			}

YES	Since no todo with id 300, error was thrown that there is none.		 "errorMessages": ["Could not find any instances with todos/300"]

YES	The GET request is received and categories is returned.		 {
			"categories": []
			}

YES	Was able to update title susing XML.	Title of id 13 now “hello”	 {
			"id": "13",
			"title": "hello",
			"doneStatus": "false",
			"description": ""
			}

YES	Server has been shutdown		 Could not get response Error: socket hang up

Summary of session findings

-The documented API functions work properly for the most part.

-Most of the error messages are intuitive

-Certain issues with curl commands which are more drilled down

-Sometimes issues arise when trying to use XML text to update objects.

Summary of Concerns

It is unusual for POST to accept a specific ID, as PUT is generally used for this purpose. The API's behavior here should be documented.

The presence of a shutdown endpoint could raise security concerns, as it potentially allows any client to stop the server.

The behavior with XML payloads should be confirmed, as it is less common in REST APIs and may not be as well-supported as JSON.

Test Ideas

Test using null parameter

Test other commands i.e. PATCH, COPY

Test commands where XML input failed with JSON

Test inputs with HTML input

Testing using computer running a different OS.