Team 6 - 2. Test cases

Deals API - Mihhail

Deals API Test Cases

Test case ID	**DSF001. Get the list of all Deals**
Description	It is possible to retrieve the full list of all existing Deals
Precondition	Some Deals exists in the application. Bearer token: b30ca93d0a7ebe389e9326a22497bf38
Test data	-
Test steps	1. Send **GET** `{baseUrl}/obj/deal`
Expected result	• Successful code: 200 OK • List of all deals is returned in response

Test case ID	**DSF002. Get the list of all Deals without authorization**
Description	It is possible to retrieve the full list of all existing Deals without authorization
Precondition	Some Deals exists in the application. Authorization token is not in use.
Test data	-
Test steps	1. Send **GET** `{baseUrl}/obj/deal`
Expected result	• 401 Permission denied • List is empty

Test case ID	**DSF003. Get Deal by its ID**
Description	It is possible to retrieve specific Deal by it's ID
Precondition	Some Deals exists in the application. Bearer token: b30ca93d0a7ebe389e9326a22497bf38
Test data	Key: UniqueID Value: 1695999864175x711716184728096600 or choose any other Deal ID using **GET** `{baseUrl}/obj/deal` request.
Test steps	1. Send **GET** using test data to `{baseUrl}/obj/deal/:UniqueID`
Expected result	• Successful code: 200 OK • Deal under specific Id is returned in response

Test case ID	**DSF004. Create a new Deal with no given data.**
Description	It is not possible to create a new deal with no given data (without required fields).
Precondition	Bearer token: b30ca93d0a7ebe389e9326a22497bf38

Test data	{ }
Test steps	1. Send **POST** `{baseUrl}/obj/deal` request using body from test data.
Expected result	• 400 Bad request. • "status": "MISSING_DATA"

POST + GET request

Test case ID	**DSF005. Create a new Deal with some given data.**
Description	It is possible to create a new deal and observe if it was created with given date by Id.
Precondition	Bearer token: b30ca93d0a7ebe389e9326a22497bf38
Test data	`{ "visible_to_list_user": ["1625051658962x581289845978797800"], "status_option_status": "In progress", "order_number": 1, "assignee__user": "1694238282399x397816817031822800", "company_name_text": "{{$randomCompanyName}}", "description_text": "{{$randomJobDescriptor}}", "deal_value_estimation_number": "{{$randomInt}}", "name_text": "{{$randomProductName}}" }`
Test steps	1. Send **POST** `{baseUrl}/obj/deal` request using body from test data. 2. Copy id of the created deal from the response. 3. Send **GET** `{baseUrl}/obj/deal/:UniqueID` , set correct UniqueID (taken from step 2).
Expected result	• Successful code: 201 Created. Deal exist • Successful code: 200 OK. Deal is found and given data exist.

Test case ID	**DSF006. Create a new Deal with some given data, but without authorization token.**
Description	It is not possible to create a new deal without authorization.
Precondition	Authorization token is not used.
Test data	`{ "visible_to_list_user": ["1625051658962x581289845978797800"], "status_option_status": "In progress", "order_number": 1, "assignee__user": "1694238282399x397816817031822800", "company_name_text": "{{$randomCompanyName}}", "description_text": "{{$randomJobDescriptor}}", "deal_value_estimation_number": "{{$randomInt}}", "name_text": "{{$randomProductName}}" }`
Test steps	1. Send **POST** `{baseUrl}/obj/deal` request using body from test data.

Expected result	• 401 Permission denied • Deal is not created

Test case ID	**DSF007. Create a new Deal with some given data, company_name_text symbol validation.**
Description	Is is not possible to create a new deal if company's name character count exceeds 100 symbols.
Precondition	Bearer token: b30ca93d0a7ebe389e9326a22497bf38
Test data	`{ "visible_to_list_user": ["1625051658962x581289845978797800"], "status_option_status": "In progress", "order_number": 1, "assignee__user": "1694238282399x397816817031822800", "company_name_text": "reeonlinecntcalculatorthassimpletouseSometimesusligreec "description_text": "{{$randomJobDescriptor}}", "deal_value_estimation_number": "{{$randomInt}}", "name_text": "{{$randomProductName}}" }`
Test steps	1. Send **POST** `{baseUrl}/obj/deal` request using body from test data.
Expected result	• 405 Wrong method

PATCH + GET request

Test case ID	**DSF008. Modify some data in the deal and check the update.**
Description	It is possible to replace a deal data and check if given data is replaced.
Precondition	Bearer token: b30ca93d0a7ebe389e9326a22497bf38
Test data	`{ "status_option_status": "Won", "company_name_text": "Some Company name", "deal_value_estimation_number": 12312313 }`
Test steps	1. Use test data to send **PATCH** `{baseUrl}/obj/deal/:UniqueID` request using ID from **DSF005** step 2. 2. Send **GET** `{baseUrl}/obj/deal/:UniqueID` , set correct UniqueID (taken from step 1).
Expected result	• Successful code: 204 No Content, update was successful • Successful code: 200 OK, correct lead is found, company and full name values were successfully updated

Test case ID	**DSF009. Modify some data in the deal without authorization.**
Description	It is not possible to replace a deal data without authorization
Precondition	authorization token is not used
Test data	`{ "status_option_status": "Lost",`

	<pre>3 "company_name_text": "Without authorization",
	4 "deal_value_estimation_number": 654321
	5 }</pre>

Test steps	1. Use test data to send **PATCH** `{baseUrl}/obj/deal/:UniqueID` request using ID from **DSF005** step 2.
Expected result	• 401 Permission denied • Deal data is not replaced

Test case ID	**DSF010. Modify some data in the deal symbol validation.**
Description	It is not possible to replace a deal data if character count exceeds 100 symbols in any field.
Precondition	Bearer token: b30ca93d0a7ebe389e9326a22497bf38

Test data	<pre>1 {
	2 "status_option_status": "In progress",
	3 "company_name_text": "reeonlinecntcalculatorthassimpletouseSometimesusligreeor
	4 "deal_value_estimation_number": 123123123123123123123123123123123123123123123
	5 }</pre>

Test steps	1. Use test data to send **PATCH** `{baseUrl}/obj/deal/:UniqueID` request using ID from **DSF005** step 2.
Expected result	• 405 Wrong method

Test case ID	**DSF011. Modify some data in the deal, mandatory fields are empty.**
Description	It is not possible to replace a deal data if mandatory fields are empty.
Precondition	Bearer token: b30ca93d0a7ebe389e9326a22497bf38

Test data	<pre>1 {
	2 "status_option_status": "",
	3 "company_name_text": "",
	4 "deal_value_estimation_number": 0
	5 }</pre>

Test steps	1. Use test data to send **PATCH** `{baseUrl}/obj/deal/:UniqueID` request using ID from **DSF005** step 2.
Expected result	• 405 Wrong method • "status": "MISSING_DATA"

PUT + GET request

Test case ID	**DSF012. Replace deal by new deal by Unique ID.**
Description	Is it possible to replace a deal by a new deal, but Unique ID stays same.
Precondition	Bearer token: b30ca93d0a7ebe389e9326a22497bf38

Test data	<pre>1 {
	2 "visible_to_list_user": [
	3 "1696156220644x114818140430645420"
	4],</pre>

	<pre>5 "status_option_status": "In progress",
	6 "order_number": 1,
	7 "assignee__user": "1625047362532x398219546419355650",
	8 "company_name_text": "Mobil1",
	9 "description_text": "Car oil",
	10 "deal_value_estimation_number": "222111222",
	11 "name_text": "name text"
	12 }</pre>

Test steps	1. Use test data to send **PUT** `{baseUrl}/obj/deal/:UniqueID` request using ID from **DSF005** step 2. 2. Send **GET** `{baseUrl}/obj/deal/:UniqueID` , set correct UniqueID (taken from step 1).
Expected result	• 204 No Content • 200 OK. Data in deal is replaced with test data.

Test case ID	**DSF013. Replace deal by new deal by Unique ID, mandatory fields are empty.**
Description	Is it possible to replace a deal by a new deal, Unique ID stays same, mandatory fields are empty.
Precondition	Bearer token: b30ca93d0a7ebe389e9326a22497bf38

Test data	<pre>1 {
	2 }</pre>

Test steps	1. Use test data to send **PUT** `{baseUrl}/obj/deal/:UniqueID` request using ID from **DSF005** step 2.
Expected result	• 405 Wrong method

Test case ID	**DSF014. Replace deal by new deal by Unique ID, without authorization.**
Description	Is it possible to replace a deal by a new deal, Unique ID stays same, without authorization.
Precondition	authorization token is not used

Test data	<pre>1 {
	2 "visible_to_list_user": [
	3 "1625051658962x581289845978797800"
	4],
	5 "status_option_status": "In progress",
	6 "order_number": 1,
	7 "assignee__user": "1625047362532x398219546419355650",
	8 "company_name_text": "Mobil1",
	9 "description_text": "Car oil",
	10 "deal_value_estimation_number": "222111222",
	11 "name_text": "name text"
	12 }</pre>

Test steps	1. Use test data to send **PUT** `{baseUrl}/obj/deal/:UniqueID` request using ID from **DSF005** step 2.
Expected result	• 401 Permission denied • Deal is not replaced

DELETE + GET request

Test case ID	**DSF015. Delete deal by Unique ID.**
Description	Is it possible to delete a deal by its Unique ID and check is it was deleted.
Precondition	Bearer token: b30ca93d0a7ebe389e9326a22497bf38
Test data	-
Test steps	1. Send **DELETE** `{baseUrl}/obj/deal/:UniqueID` request using ID from **DSF005** step 2. 2. Send **GET** `{baseUrl}/obj/deal/:UniqueID` , set correct UniqueID (used in step 1).
Expected result	• 204 No Content • 404 Not Found. "status": "MISSING_DATA", "message": "Missing object of type deal: object with id from step 1 does not exist"

Test case ID	**DSF016. Delete deal by Unique ID without authorization.**
Description	Is it possible to delete a deal by its Unique ID without authorization token.
Precondition	authorization token is not used
Test data	-
Test steps	1. Send **DELETE** `{baseUrl}/obj/deal/:UniqueID` request using ID from **DSF005** step 2.
Expected result	• 401 Permission denied • Deal is not deleted

Notes API - Valeri

Notes API Test Cases

GET

Test case ID	**NSF001. Get the list of all Notes**
Description	Check that it is possible to retrieve the list of all existing notes
Precondition	Some notes exists in the application
Test data	-
Test steps	1. Send **GET** `{baseUrl}/obj/note`
Expected result	• Successful code: 200 OK • List of all notes is returned in response

Test case ID	**NSF002. Get the Note by ID**
Description	Check that it is possible to retrieve the data of a note by its unique ID
Precondition	Some notes exists in the application
Test data	UniqueID : 1695749265428x206466228315226100

Test steps	1. Send **GET** `{baseUrl}/obj/note/:UniqueID`
Expected result	• Successful code: 200 OK • API returns the data of a note by its unique ID

Test case ID	**NSF003. Sort the Notes in descending order**
Description	Check that it is possible to sort the notes in descending order
Precondition	Some notes exists in the application
Test data	Set parameters **sort_field** - **content_text** and **descending** - **true**
Test steps	1. Send **GET** `{baseUrl}/obj/note?sort_field=content_text&descending=true`
Expected result	• Successful code: 200 OK • The notes which is returned in response are displayed in descending order

Test case ID	**NSF004. Get the Note with specific text**
Description	Check that it is possible to retrieve the note with specific text
Precondition	Some notes exists in the application
Test data	-
Test steps	1. Send **GET** `{baseUrl}/obj/note?constraints=[ { "key": "content_text", "constraint_type": "equals", "value": "Note from Val : I added this note to the Deal using Postman" }]`
Expected result	• Successful code: 200 OK • The notes with specific text is returned in response

Test case ID	**NSF005. Get the list of Notes with parameter cursor**
Description	Check that it is possible to retrieve the list of existing notes with parameter cursor
Precondition	Some notes exists in the application
Test data	-
Test steps	1. Send **GET** `{baseUrl}/obj/note?cursor=15`
Expected result	• Successful code: 200 OK • List of notes is returned in response from position 15

Test case ID	**NSF006. Get the list of Notes with negative value in parameter cursor**
Description	Check that it is not possible to retrieve the list of existing notes with negative value in parameter cursor
Precondition	Some notes exists in the application
Test data	-
Test steps	1. Send **GET** `{baseUrl}/obj/note?cursor=-1`

Expected result	• Error code: 500 Internal Server Error • List of all notes is not returned in response, error message is shown `"Sorry, we ran into a temporary bug and can't complete your request. We'll fix it as soon as we can; please try again in a bit!"`

Test case ID	**NSF007. Get the list of Notes with parameter limit**
Description	Check that it is possible to retrieve the list of notes with parameter limit (*Default value* : 50)
Precondition	50+ (at least 51) notes must be exists in the application
Test data	-
Test steps	1. Send **GET** `{baseUrl}/obj/note?limit=0`
Expected result	• Successful code: 200 OK • The list of 50 notes is returned in response

Test case ID	**NSF008. Get the list of Notes with parameter limit as string type**
Description	Check that it is not possible to retrieve the list of notes with parameter limit as a string type
Precondition	Some notes exists in the application
Test data	-
Test steps	1. Send **GET** `{baseUrl}/obj/note?limit=asd`
Expected result	• List of notes is not returned in response • An error code is displayed and error message is shown

Test case ID	**NSF009. Get the Note with negative value in parameter limit**
Description	Check that it is not possible to retrieve the list of notes with negative value in parameter limit
Precondition	Some notes exists in the application
Test data	-
Test steps	1. Send **GET** `{baseUrl}/obj/note?limit=-1`
Expected result	• Successful code: 200 OK • An empty list is returned in response

Test case ID	**NSF010. Get the list of Notes with parameter limit as decimal**
Description	Check that it is not possible to retrieve the list of notes with parameter limit as decimal
Precondition	Some notes exists in the application
Test data	-
Test steps	1. Send **GET** `{baseUrl}/obj/note?limit=1.5`
Expected result	• List of notes is not returned in response

	• An error code is displayed and error message is shown

Test case ID	**NSF011. Search for a Note with a non-existent content text**
Description	Check that an empty list of results is retuned when a non-existent text is used as a filter
Precondition	Note with given text doesn't exist
Test data	-
Test steps	1. Send **GET** `{baseUrl}/obj/note?constraints=[ { "key": "content_text", "constraint_type": "equals", "value": "Val non-existing note" }]`
Expected result	• Successful code: 200 OK • Empty list is returned in response

POST + GET request

Test case ID	**NSF012. Create Note and check the created object**
Description	Check that it is possible to create a new note and observe if it is was created successfully with given data
Precondition	Bearer token, aka authToken: `b30ca93d0a7ebe389e9326a22497bf38`

Test data	`{`
	`  "content_text": "My Test Note Team 6"`
	`}`

Test steps	1. Send **POST** `{baseUrl}/obj/note` request using body from test data 2. Copy ID of the created note from the response 3. Send **GET** `{baseUrl}/obj/note/:UniqueID`, set correct UniqueID (taken from step 2)
Expected result	• Successful code: 201 Created, note is created • Successful code: 200 OK, correct note is found, API returns the data of a note by its unique Id

Test case ID	**NSF013. Create Note with empty body**
Description	Check that it is not possible to create a new note without `content_text` field(mandatory field)
Precondition	Bearer token, aka authToken: `b30ca93d0a7ebe389e9326a22497bf38`

Test data	`{`
	
	`}`

Test steps	1. Send **POST** `{baseUrl}/obj/note` request using body from test data
Expected result	• Error code: 400 Bad Request • Object is not created, error message is shown `"message": "NO_CONTENT_TEXT", "translation": "Please include an text"`

Test case ID	**NSF014. Create Note without authorization key**
Description	Check that it is not possible to create a new note without authorization key
Precondition	Authorization - No Auth, authorization key is not used
Test data	`{ "content_text": "My Test Note Team 6" }`
Test steps	1. Send **POST** `{baseUrl}/obj/note` request using body from test data
Expected result	• Error code: 401 Unauthorized • Object is not created, error message is shown `"message": null, "translation": "Permission denied: cannot create this object"`

PATCH + GET request

Test case ID	**NFS015. Update Note and check the data**
Description	Check that it is possible to update existing Note and observe if it is was updated successfully with given data
Precondition	• **NSF012** successfully completed • Bearer token, aka authToken: `b30ca93d0a7ebe389e9326a22497bf38`
Test data	`{ "content_text": "My Patched Note Team 6" }`
Test steps	1. Send **PATCH** `{baseUrl}/obj/note/:UniqueID` , set correct UniqueID (taken from step 2, **NSF012**) 2. Send **GET** `{baseUrl}/obj/note/:UniqueID` , set correct UniqueID (taken from previous step)
Expected result	• Successful code: 204 No Content, update was successful • Successful code: 200 OK, correct note is found, note values were successfully updated

Test case ID	**NSF016. Update Note without authorization key**
Description	Check that it is not possible to update a note without authorization key
Precondition	• **NSF012** successfully completed • Authorization - No Auth, authorization key is not used
Test data	-
Test steps	1. Send **PATCH** `{baseUrl}/obj/note/:UniqueID` , set correct UniqueID (taken from step 2, **NSF012**)
Expected result	• Error code: 401 Unauthorized • Object is not created, error message is shown `"message": null, "translation": "You do not have permission to modify this object"`

PUT + GET request

Test case ID	**NSF017. Replace Note and check the data**
Description	Check that it is possible to replace Note and observe if it is was replaced successfully with given data
Precondition	• **NSF012** successfully completed • Bearer token, aka authToken: `b30ca93d0a7ebe389e9326a22497bf38`
Test data	`{ "content_text": "My Replaced Note Team 6" }`
Test steps	1. Send **PUT** `{baseUrl}/obj/note/:UniqueID` , set correct UniqueID (taken from step 2, **NSF012**) 2. Send **GET** `{baseUrl}/obj/note/:UniqueID`, set correct UniqueID (taken from previous step)
Expected result	• Successful code: 204 No Content, replacement was successful • Successful code: 200 OK, correct note is found, note values were successfully replaced

Test case ID	**NSF018. Replace Note without authorization key**
Description	Check that it is not possible to replace Note and without authorization key
Precondition	Authorization - No Auth, authorization key is not used
Test data	`{ "content_text": "My Replaced Note Team 6" }`
Test steps	1. Send **PUT** `{baseUrl}/obj/note/:UniqueID` , set correct UniqueID (taken from step 2, **NSF012**) 2. Send **GET** `{baseUrl}/obj/note/:UniqueID`, set correct UniqueID (taken from previous step)
Expected result	• Error code: 401 Unauthorized • Object is not created, error message is shown `"message": null, "translation": "You do not have permission to modify this object"`

DELETE + GET request

Test case ID	**NFS019. Delete Note and check data**
Description	Check that it is possible to delete existing note
Precondition	• **NSF012** successfully completed • Bearer token, aka authToken: `b30ca93d0a7ebe389e9326a22497bf38`
Test data	-
Test steps	1. Send **DELETE** `{baseUrl}/obj/note/:UniqueID` , set correct UniqueID (taken from step 2, **NSF012**) 2. Send **GET** `{baseUrl}/obj/note/:UniqueID`, set correct UniqueID (taken from previous step)
Expected result	• Successful code: 204 No Content, delete was successful • Error code: 404 Not Found, `"status": "MISSING_DATA", "message": "Missing object of type note: object with id {{UniqueId}} does not exist"`

Test case ID	**NSF020. Delete Note without authorization key**
Description	Check that it is not possible to delete a note without authorization key
Precondition	Authorization - No Auth, authorization key is not used
Test data	-
Test steps	1. Send **DELETE** `{baseUrl}/obj/note/:UniqueID` , set correct UniqueID (taken from step 2, **NSF012**)
Expected result	• Error code: 401 Unauthorized • Object is not created, error message is shown `"message": null, "translation": "You do not have permission to delete this object"`