

# Mind Teams Challenge

Brayan Prieto Lozoya – Developer

Daniel León Baro – QA

1. Log In.....	5
1.1. Test Plan.....	5
1.2. Test Cases.....	7
<i>Negative Scenario</i> : Try accessing with a wrong username.....	7
<i>Negative Scenario</i> : Try accessing with a wrong password .....	7
<i>Happy Path</i> : Try accessing with a valid username.....	7
<i>Negative Scenario</i> : Try creating a user with a username already used.....	8
<i>Happy Path</i> : Verify that Passwords are encrypted in the DB .....	8
<i>API Test</i> : Generate a Bearer Token for Login.....	8
1.3. Bugs and Improvements .....	10
2. Log out .....	11
2.1. Test Plan.....	11
2.2. Test Cases.....	12
<i>Happy Path</i> : Try logging out from a Super User account .....	12
<i>Happy Path</i> : Try logging out from an Admin account.....	12
<i>Happy Path</i> : Try logging out from a User account.....	12
2.3. Bugs and Improvements .....	14
3. Role Management.....	15
3.1. Test Plan.....	15
3.2. Test Cases.....	17
<i>API Test</i> : Verify Get Users API Request works as expected .....	17
<i>API Test</i> : Verify Get User API Request works as expected.....	17
<i>API Test</i> : Verify Create User API Request works as expected.....	18
<i>API Test</i> : Verify Modify User API Request works as expected .....	19
<i>API Test</i> : Verify Delete User API Request works as expected.....	19
<i>API Test</i> : Verify Get Profile API Request works as expected.....	20
<i>API Test</i> : Verify Get Teams API Request works as expected.....	20
<i>API Test</i> : Verify Create Teams API Request works as expected.....	21
<i>API Test</i> : Verify Get Team API Request works as expected .....	21
<i>API Test</i> : Verify Delete Team API Request works as expected .....	22
<i>API Test</i> : Verify Modify Teams API Request works as expected.....	22

<i>API Test</i> : Verify Get Teams API Request works as expected.....	23
<i>API Test</i> : Verify Create Teams API Request works as expected.....	23
<i>API Test</i> : Verify Get Team API Request works as expected .....	24
<i>API Test</i> : Verify Delete Team API Request works as expected .....	24
<i>API Test</i> : Verify Modify Teams API Request works as expected .....	25
<i>API Test</i> : Verify Get Accounts API Request works as expected .....	25
<i>API Test</i> : Verify Create Account API Request works as expected .....	26
<i>API Test</i> : Verify Get Account API Request works as expected .....	26
<i>API Test</i> : Verify Delete Account API Request works as expected .....	27
<i>API Test</i> : Verify Modify Account API Request works as expected .....	27
<i>Happy Path</i> : Verify you can see the User CRUD when logging in as an Admin or Super User .....	28
<i>Happy Path</i> : Verify you can create a User in the User CRUD when logging in as an Admin or Super User .....	28
<i>Happy Path</i> : Verify you can see Users in the User CRUD when logging in as an Admin or Super User .....	28
<i>Happy Path</i> : Verify you can search a user in the User CRUD when logging in as an Admin or Super User .....	29
<i>Happy Path</i> : Verify you can delete a user in the User CRUD when logging in as an Admin or Super User .....	29
<i>Happy Path</i> : Verify you can see the Teams CRUD when logging in as an Admin or Super User .....	30
<i>Happy Path</i> : Verify you can create a team in the Teams CRUD when logging in as an Admin or Super User .....	30
<i>Negative Scenario</i> : Verify you can't create a user in the Users CRUD without a Name.....	30
<i>Negative Scenario</i> : Verify you can't create a user in the Users CRUD with an invalid Email .....	31
<i>Negative Scenario</i> : Verify you can't create a user in the Users CRUD without a password.....	31
<i>Negative Scenario</i> : Verify you can't create a user in the Users CRUD without a CV .....	32
<i>Negative Scenario</i> : Verify you can't create a user in the Users CRUD without Experience .....	32
<i>Negative Scenario</i> : Verify you can't create a user in the Users CRUD without an English level .....	32
<i>Negative Scenario</i> : Verify you can't create a user in the Users CRUD with a different Password in the confirmation.....	33
<i>Happy Path</i> : Verify you can see teams in the Teams CRUD when logging in as an Admin or Super User .....	33
<i>Happy Path</i> : Verify you can search a team in the Teams CRUD when logging in as an Admin or Super User .....	34
<i>Happy Path</i> : Verify you can delete a team in the Teams CRUD when logging in as an Admin or Super User .....	34
<i>Negative Scenario</i> : Verify you can't create a team in the Teams CRUD without a Name.....	34

<i>Negative Scenario</i> : Verify you can't create a team in the Teams CRUD without a Description .....	35
<i>Happy Path</i> : Verify you can see the Accounts CRUD when logging in as an Admin or Super User ....	35
<i>Happy Path</i> : Verify you can create an account in the Accounts CRUD when logging in as an Admin or Super User .....	36
<i>Negative Scenario</i> : Verify you can't create an account in the Accounts CRUD without a Name .....	36
<i>Negative Scenario</i> : Verify you can't create an account in the Accounts CRUD without a Customer .	36
<i>Negative Scenario</i> : Verify you can't create an account in the Accounts CRUD without a Manager ..	37
<i>Negative Scenario</i> : Verify you can't create an account in the Accounts CRUD without a Team .....	37
<i>Happy Path</i> : Verify you can see accounts in the Accounts CRUD when logging in as an Admin or Super User .....	38
<i>Happy Path</i> : Verify you can search an account in the Accounts CRUD when logging in as an Admin or Super User .....	38
<i>Happy Path</i> : Verify you can delete an account in the Accounts CRUD when logging in as an Admin or Super User .....	38
<i>Happy Path</i> : Verify you can see the Operations CRUD when logging in as an Admin or Super User .	39
<i>Happy Path</i> : Verify you can View History in the Operations CRUD when logging in as an Admin or Super User .....	39
<i>Happy Path</i> : Verify you can View History by dates in the Operations CRUD when logging in as an Admin or Super User .....	40
<i>Happy Path</i> : Verify you can View History by user in the Operations CRUD when logging in as an Admin or Super User .....	40
<i>Happy Path</i> : Verify you can View History by team in the Operations CRUD when logging in as an Admin or Super User .....	40
<i>Happy Path</i> : Verify you can move users between teams in the Operations CRUD when logging in as an Admin or Super User .....	41
3.3. Bugs and Improvements .....	42
You are able to Search by ID but you get an error in console .....	42
You are not able to Search by Name .....	42
You get to a blank page if you search non-existence ID .....	43
View History, View History by User, and View History by Team does not store the dates .....	44
View History by Dates does not change .....	45
View History by User randomly fills User as "None" .....	45
Credentials still logged in after removing user from DB .....	46
Delete CRUD does not exist .....	46
Create Account API response returns a null Team value that is not needed .....	47
Cancel button says cancel on Lists .....	47



# Stories

## 1. Log In

### 1.1. Test Plan

#### Description

1. What is the story trying to solve?

There is no way to log in into the platform

2. What is the proposed solution?

A form for the user to log in and be able to enter the User Management CRUD

3. How is the solution being implemented?

A dedicated page for the user to log in credentials and get in the platform, depending on the user type to log in a different view

#### Risk Analysis

<b>Risk:</b> Passwords might not be encrypted	<b>Tolerance:</b> Transferred
Verify in the Database that passwords are encrypted	If the risk happens, it will be reassigned to the developer team.

<b>Risk:</b> Log in form might not be Case Sensitive	<b>Tolerance:</b> Accepted
Test using a log in credential that originally is upper case and inputting them as lower case	If risk happens, no actions will be taken.

<b>Risk:</b> Duplicated Log Ins	<b>Tolerance:</b> Avoid
Test trying to access an account with duplicated information, same user and same password	If the risk happens, it will be reassigned to the developer team.

#### Testing Steps

Based on Criteria:

1. Verify Home page is the Log In credentials if no user was logged in before
2. Verify Home page is the home page if there was a user already logged in

Based on Risk Analysis:

1. Verify you can't create duplicated Super Admin credentials
  - a. Verify that you can't log in in the duplicated credentials
2. Verify trying to log in to a "User1" using "user1"
3. Verify that when creating a Super User, password is encrypted
4. Verify that when creating an Admin, password is encrypted
5. Verify that when creating a User, password is encrypted

## 1.2. Test Cases

Negative Scenario: Try accessing with a wrong username

### Description:

Verify that we get an error when trying to access with a not existing username

### Steps:

1. Enter the Home Page
2. Fill a random string in the Username input making sure it doesn't exist as a user before, for instance, "abcd"
3. Fill a random string in the Password input
4. Click on Log In

### Test Validation:

If you get an error advising that the username does not exist, the test passes.

Negative Scenario: Try accessing with a wrong password

### Description:

Verify that we get an error when trying to access with a wrong password in an existing user

### Steps:

1. Enter the Home Page
2. Fill an existing Username in the Username input
3. Fill a random string in the Password input that is not the correct password of the User
4. Click on Log In

### Test Validation:

If you get an error advising that the password is incorrect, the test passes.

Happy Path: Try accessing with a valid username

### Description:

Verify that we get successfully log in using existing credentials

### Steps:

1. Enter the Home Page
2. Fill both Username and Password credentials in their corresponding inputs
3. Click on Log In

### Test Validation:

If you are redirected to the CRUD if the user is Admin or Super User the test passes.

If you are redirected to your profile if the user is a common User the test passes.

Negative Scenario: Try creating a user with a username already used

**Description:**

Verify that we get an error when trying to create a user with the same name

**Steps:**

1. Login as an Admin
2. Enter the Create User menu
3. Create a user with a name, for instance, UserTest
4. Try creating a new user with the name used in step 3

**Test Validation:**

If you get an error advising that the username already exist, the test passes.

Happy Path: Verify that Passwords are encrypted in the DB

**Description:**

Verify that when we create a fresh User, the password is hashed in the DB

**Steps:**

1. Create a Super User directly in the DB
2. Login as a Super User
3. Enter the Create User menu
4. Create a user with a name, for instance, UserEncrypted
5. Create an admin with a name, for instance, AdminEncrypted
6. Enter the DB

**Test Validation:**

If you see the three passwords from the SuperUser, the Admin and the User as hashed, the test passes.

API Test: Generate a Bearer Token for Login

**Description:**

Verify that you are able to login with valid credentials and get a Bearer Token through the API

**Steps:**

1. Open Postman
2. Generate a new GET request
3. Put in the URL: <https://localhost:44309/api/Users/login>



4. In the Body, use this JSON using SuperUser credentials:

```
{  
  "name": "(Insert a SuperUser name account)",  
  "password": "(Insert a SuperUser password)"  
}
```

5. Send the request

**Test Validation:**

If get a token in the Response, the test passes.

### 1.3. Bugs and Improvements

None yet

## 2. Log out

### 2.1. Test Plan

#### Description

1. What is the story trying to solve?

There is no way for the user to log out of the platform and be able to re-login with the same credentials or different ones

2. What is the proposed solution?

A way to log out from the platform

3. How is the solution being implemented?

A button that allows the user to log out

#### Risk Analysis

<b>Risk:</b> Credentials might stay after logging out	<b>Tolerance:</b> Avoid
Testing that the user is not logged in when you click in the Log Out button	If risk happens, it will be re-assigned to the developer team.

#### Testing Steps

##### Based on Criteria:

1. When logged in, verify you can see a Log Out button
2. When logged out, verify you cannot see a Log Out button
3. Verify that clicking Log Out, logs you out

##### Based on Risk Analysis:

1. Verify that after clicking Log Out, reloading the page won't log you in automatically with the previous credentials

## 2.2. Test Cases

*Happy Path:* Try logging out from a Super User account

**Description:**

We try logging out from the platform successfully being logged in as a Super User

**Steps:**

1. Enter the Home Page
2. Fill both Username and Password credentials in their corresponding inputs, using Super User credentials
3. Click on Log In
4. Once logged in, click on the Log out button

**Test Validation:**

If you're logged out successfully, the test passes.

*Happy Path:* Try logging out from an Admin account

**Description:**

We try logging out from the platform successfully being logged in as an Admin

**Steps:**

1. Enter the Home Page
2. Fill both Username and Password credentials in their corresponding inputs, using Admin credentials
3. Click on Log In
4. Once logged in, click on the Log out button

**Test Validation:**

If you're logged out successfully, the test passes.

*Happy Path:* Try logging out from a User account

**Description:**

We try logging out from the platform successfully being logged in as a User

**Steps:**

1. Enter the Home Page
2. Fill both Username and Password credentials in their corresponding inputs, using User credentials
3. Click on Log In
4. Once logged in, click on the Log out button

**Test Validation:**

If you're logged out successfully, the test passes.

### 2.3. Bugs and Improvements

None yet

## 3. Role Management

### 3.1. Test Plan

#### Description

1. What is the story trying to solve?

There is no actual way to create and modify users

2. What is the proposed solution?

A page that depends on the user, if the user is an Admin or Super User, it allows to modify the account type of an existing user and create users

3. How is the solution being implemented?

A CRUD that shows if the user is an Admin or Super User, and information consultation if the user is a User

#### Risk Analysis

<b>Risk:</b> SU (Super User) might be able to be created in the CRUD	<b>Tolerance:</b> Avoid
Test if you can create a SU in the CRUD instead of the database	If the risk happens, it will be reassigned to the developer team.

<b>Risk:</b> SU (Super User) might be able to be created via the API	<b>Tolerance:</b> Avoid
Test if you can create a SU with the API instead of the database	If the risk happens, it will be reassigned to the developer team.

<b>Risk:</b> Admins might be able to create more Admins	<b>Tolerance:</b> Avoid
Test you can only create more Admins with a SU	If the risk happens, it will be reassigned to the developer team.

#### Testing Steps

Based on Criteria:

2. [API Test](#): GET Users
3. [API Test](#): POST Users
4. [API Test](#): GET Users with ID as parameter

5. *API Test*: PUT Users by ID as parameter
6. *API Test*: DELETE Users by ID as parameter
7. Verify that logging in with a Super User logs you to the Super User CRUD
8. Verify that logging in with an Admin logs you to the Admin CRUD
9. Verify that logging in with a User logs you to the User Profile
10. Verify that as a User you can only see your profile
- 10.1. Verify that as a User you cannot modify your profile
- 10.2. Verify that as a User you cannot create more users
11. Verify that as an Admin and Super User you can access the CRUD
- 11.1. Verify you can create Users with Name, Mail and Encrypted Password
- 11.2. Verify that you can assign Accounts to the User with Account Name, Client Name, Responsible of Operations, Consultations of team
- 11.3. Verify that as an Admin and Super User you can consult the Movements Logs and see Start date of an account, End date, and that it keeps the User information of that account
- 11.4. Verify that you can consult all user's information

Based on Risk Analysis:

1. Ipsum
2. Ipsum
- 2.1. Ipsum Lorem



### 3.2. Test Cases

*API Test:* Verify Get Users API Request works as expected

**Description:**

Verify by using the Get Users request through the API without a parameter responds a list of the users available in the database.

**Steps:**

1. Open Postman
2. Generate a new GET request
3. Put in the URL: <https://localhost:44309/api/Users/login>
4. In the Body, use this JSON using SuperUser credentials:

```
{
  "name": "(Insert a SuperUser name account)",
  "password": "(Insert a SuperUser password)"
}
```
5. Send the request
6. Copy the token in the response
7. Generate a new GET request
8. Put in the URL: <https://localhost:44309/api/Users/all>
9. Enter Auth and select Type Bearer Token
10. Paste the token in the Token input
11. Send the request

**Test Validation:**

If get a list of the available users, the test passes.

*API Test:* Verify Get User API Request works as expected

**Description:**

Verify by using the Get User request through the API without a parameter responds with the information of a specific user in the database.

**Steps:**

1. Open Postman
2. Generate a new GET request
3. Put in the URL: <https://localhost:44309/api/Users/login>
4. In the Body, use this JSON using SuperUser credentials:

```
{
  "name": "(Insert a SuperUser name account)",
  "password": "(Insert a SuperUser password)"
}
```
5. Send the request

6. Copy the token in the response
7. Generate a new GET request
8. Put in the URL: <https://localhost:44309/api/Users/{UserID}> of the customer you want to view)
9. Enter Auth and select Type Bearer Token
10. Paste the token in the Token input
11. Send the request

**Test Validation:**

If get a the user information, the test passes.

*API Test:* Verify Create User API Request works as expected

**Description:**

Verify by using the Create User request through the API allows you to create a User.

**Steps:**

1. Open Postman
2. Generate a new GET request
3. Put in the URL: <https://localhost:44309/api/Users/login>
4. In the Body, use this JSON using SuperUser credentials:

```
{
  "name": "(Insert a SuperUser name account)",
  "password": "(Insert a SuperUser password)"
}
```
5. Send the request
6. Copy the token in the response
7. Generate a new POST request
8. Put in the URL: <https://localhost:44309/api/Users>
9. Enter Auth and select Type Bearer Token
10. Paste the token in the Token input
11. In Body, paste the next:

```
{
  "name": "(User name you want)",
  "password": "(Password you want, needs 8 characters and at least one Upper case character and a number)"
}
```
12. Send the request

**Test Validation:**

If get a 200 response, the test passes.

**API Test:** Verify Modify User API Request works as expected

**Description:**

Verify by using the Modify User request through the API allows you to modify a User.

**Steps:**

1. Open Postman
2. Generate a new GET request
3. Put in the URL: <https://localhost:44309/api/Users/login>
4. In the Body, use this JSON using SuperUser credentials:

```
{
  "name": "(Insert a SuperUser name account)",
  "password": "(Insert a SuperUser password)"
}
```
5. Send the request
6. Copy the token in the response
7. Generate a new PUT request
8. Put in the URL: [https://localhost:44309/api/Users/\(UserID\)](https://localhost:44309/api/Users/(UserID)) of the customer you want to modify)
9. Enter Auth and select Type Bearer Token
10. Paste the token in the Token input
11. In Body, paste the next:

```
{
  "name": "(User name you want)",
  "password": "(Password you want, needs 8 characters and at least one Upper case character
and a number)"
}
```
12. Send the request

**Test Validation:**

If get a 200 response, the test passes.

**API Test:** Verify Delete User API Request works as expected

**Description:**

Verify by using the Delete User request through the API allows you to remove a User from the DB.

**Steps:**

1. Open Postman
2. Generate a new GET request
3. Put in the URL: <https://localhost:44309/api/Users/login>
4. In the Body, use this JSON using SuperUser credentials:

```
{
  "name": "(Insert a SuperUser name account)",
  "password": "(Insert a SuperUser password)"
}
```

- ```
}
```
5. Send the request
  6. Copy the token in the response
  7. Generate a new DELETE request
  8. Put in the URL: <https://localhost:44309/api/Users/{UserID}> of the customer you want to delete)
  9. Enter Auth and select Type Bearer Token
  10. Paste the token in the Token input
  11. Send the request

**Test Validation:**

If get a 200 response, the test passes.

*API Test:* Verify Get Profile API Request works as expected

**Description:**

Verify by using the Get Profile request through the API without a parameter responds with the information of a specific user in the database I logged in with.

**Steps:**

1. Open Postman
2. Generate a new GET request
3. Put in the URL: <https://localhost:44309/api/Users/login>
4. In the Body, use this JSON using SuperUser credentials:

```
{
  "name": "(Insert a SuperUser name account)",
  "password": "(Insert a SuperUser password)"
}
```
5. Send the request
6. Copy the token in the response
7. Generate a new GET request
8. Put in the URL: <https://localhost:44309/api/Users/getProfile>
9. Enter Auth and select Type Bearer Token
10. Paste the token in the Token input
11. Send the request

**Test Validation:**

If you get the profile from the user you logged in, the test passes.

*API Test:* Verify Get Teams API Request works as expected

**Description:**

Verify by using the Get Teams request through the API without a parameter responds with the information of all the teams stored in the DB.

**Steps:**

1. Open Postman
2. Send the Generate Bearer Token request
3. Generate a new GET request
4. Put in the URL: <https://localhost:44309/api/Teams>
5. Enter Auth and select Type Bearer Token
6. Select the variable BearerToken
7. Send the request

**Test Validation:**

If you get a list of all the teams, the test passes.

*API Test:* Verify Create Teams API Request works as expected

**Description:**

Verify by using the Create Teams request through the API without a parameter creates successfully a Team in the DB.

**Steps:**

1. Open Postman
2. Send the Generate Bearer Token request
3. Generate a new POST request
4. Put in the URL: <https://localhost:44309/api/Teams>
5. Enter Auth and select Type Bearer Token
6. Select the variable BearerToken
7. Go to Body and fill the next JSON:

```
{
  "name": "Team Name",
  "description": "Team Description"
}
```
8. Send the request

**Test Validation:**

If you get the information of the created team and it gets stored in the DB, the test passes.

*API Test:* Verify Get Team API Request works as expected

**Description:**

Verify by using the Get Team request through the API with a parameter returns the information of the specified team.

**Steps:**

1. Open Postman
2. Send the Generate Bearer Token request
3. Generate a new GET request
4. Put in the URL: [https://localhost:44309/api/Teams/\(teamId\)](https://localhost:44309/api/Teams/(teamId))
5. Enter Auth and select Type Bearer Token
6. Select the variable BearerToken
7. Send the request

**Test Validation:**

If you get the information of the specified team, the test passes.

*API Test:* Verify Delete Team API Request works as expected

**Description:**

Verify by using the Delete Team request through the API removes a Team from the DB.

**Steps:**

1. Open Postman
2. Send the Generate Bearer Token request
3. Generate a new DELETE request
4. Put in the URL: [https://localhost:44309/api/Teams/\(teamId\)](https://localhost:44309/api/Teams/(teamId))
5. Enter Auth and select Type Bearer Token
6. Select the variable BearerToken
7. Send the request

**Test Validation:**

If the specified team gets removed from the DB, the test passes.

*API Test:* Verify Modify Teams API Request works as expected

**Description:**

Verify by using the Modify Teams request through the API without a parameter modifies successfully a Team in the DB.

**Steps:**

1. Open Postman
2. Send the Generate Bearer Token request
3. Generate a new PUT request

4. Put in the URL: <https://localhost:44309/api/Teams>
5. Enter Auth and select Type Bearer Token
6. Select the variable BearerToken
7. Go to Body and fill the next JSON:

```
{
  "name": "Team Name",
  "description": "Team Description"
}
```
8. Send the request

**Test Validation:**

If you get the information of the created team and it gets stored in the DB, the test passes.

*API Test:* Verify Get Teams API Request works as expected

**Description:**

Verify by using the Get Teams request through the API without a parameter responds with the information of all the teams stored in the DB.

**Steps:**

8. Open Postman
9. Send the Generate Bearer Token request
10. Generate a new GET request
11. Put in the URL: <https://localhost:44309/api/Teams>
12. Enter Auth and select Type Bearer Token
13. Select the variable BearerToken
14. Send the request

**Test Validation:**

If you get a list of all the teams, the test passes.

*API Test:* Verify Create Teams API Request works as expected

**Description:**

Verify by using the Create Teams request through the API without a parameter creates successfully a Team in the DB.

**Steps:**

9. Open Postman
10. Send the Generate Bearer Token request
11. Generate a new POST request
12. Put in the URL: <https://localhost:44309/api/Teams>

13. Enter Auth and select Type Bearer Token
14. Select the variable BearerToken
15. Go to Body and fill the next JSON:

```
{
  "name": "Team Name",
  "description": "Team Description"
}
```
16. Send the request

**Test Validation:**

If you get the information of the created team and it gets stored in the DB, the test passes.

*API Test:* Verify Get Team API Request works as expected

**Description:**

Verify by using the Get Team request through the API with a parameter returns the information of the specified team.

**Steps:**

8. Open Postman
9. Send the Generate Bearer Token request
10. Generate a new GET request
11. Put in the URL: <https://localhost:44309/api/Teams/{teamId}>
12. Enter Auth and select Type Bearer Token
13. Select the variable BearerToken
14. Send the request

**Test Validation:**

If you get the information of the specified team, the test passes.

*API Test:* Verify Delete Team API Request works as expected

**Description:**

Verify by using the Delete Team request through the API removes a Team from the DB.

**Steps:**

8. Open Postman
9. Send the Generate Bearer Token request
10. Generate a new DELETE request
11. Put in the URL: <https://localhost:44309/api/Teams/{teamId}>
12. Enter Auth and select Type Bearer Token
13. Select the variable BearerToken



14. Send the request

**Test Validation:**

If the specified team gets removed from the DB, the test passes.

*API Test:* Verify Modify Teams API Request works as expected

**Description:**

Verify by using the Modify Teams request through the API without a parameter modifies successfully a Team in the DB.

**Steps:**

9. Open Postman
10. Send the Generate Bearer Token request
11. Generate a new PUT request
12. Put in the URL: <https://localhost:44309/api/Teams>
13. Enter Auth and select Type Bearer Token
14. Select the variable BearerToken
15. Go to Body and fill the next JSON:

```
{
  "name": "Team Name",
  "description": "Team Description"
}
```
16. Send the request

**Test Validation:**

If you get the information of the created team and it gets stored in the DB, the test passes.

*API Test:* Verify Get Accounts API Request works as expected

**Description:**

Verify by using the Get Accounts request through the API without a parameter responds with the information of all the accounts stored in the DB.

**Steps:**

15. Open Postman
16. Send the Generate Bearer Token request
17. Generate a new GET request
18. Put in the URL: <https://localhost:44309/api/Accounts>
19. Enter Auth and select Type Bearer Token
20. Select the variable BearerToken
21. Send the request

**Test Validation:**

If you get a list of all the accounts, the test passes.

*API Test:* Verify Create Account API Request works as expected

**Description:**

Verify by using the Create Account request through the API without a parameter creates successfully a Team in the DB.

**Steps:**

17. Open Postman
18. Send the Generate Bearer Token request
19. Generate a new POST request
20. Put in the URL: <https://localhost:44309/api/Account>
21. Enter Auth and select Type Bearer Token
22. Select the variable BearerToken
23. Go to Body and fill the next JSON:

```
{
  "accountName": "account",
  "customerName": "customer",
  "operationManagerName": "manager",
  "teamId": teamId
}
```
24. Send the request

**Test Validation:**

If you get the information of the created account and it gets stored in the DB, the test passes.

*API Test:* Verify Get Account API Request works as expected

**Description:**

Verify by using the Get Account request through the API with a parameter returns the information of the specified account.

**Steps:**

15. Open Postman
16. Send the Generate Bearer Token request
17. Generate a new GET request
18. Put in the URL: <https://localhost:44309/api/Account/{accountId}>
19. Enter Auth and select Type Bearer Token
20. Select the variable BearerToken
21. Send the request

**Test Validation:**

If you get the information of the specified account, the test passes.

*API Test:* Verify Delete Account API Request works as expected

**Description:**

Verify by using the Delete Account request through the API removes an Account from the DB.

**Steps:**

15. Open Postman
16. Send the Generate Bearer Token request
17. Generate a new DELETE request
18. Put in the URL: <https://localhost:44309/api/Account/{accountId}>
19. Enter Auth and select Type Bearer Token
20. Select the variable BearerToken
21. Send the request

**Test Validation:**

If the specified Account gets removed from the DB, the test passes.

*API Test:* Verify Modify Account API Request works as expected

**Description:**

Verify by using the Modify Account request through the API without a parameter modifies successfully an Account in the DB.

**Steps:**

17. Open Postman
18. Send the Generate Bearer Token request
19. Generate a new PUT request
20. Put in the URL: <https://localhost:44309/api/Teams>
21. Enter Auth and select Type Bearer Token
22. Select the variable BearerToken
25. Go to Body and fill the next JSON:

```
{
  "accountName": "account",
  "customerName": "customer",
  "operationManagerName": "manager",
  "teamId": teamId
}
```
23. Send the request

**Test Validation:**

If you get the information of the modified account and it gets stored in the DB, the test passes.

Happy Path: Verify you can see the User CRUD when logging in as an Admin or Super User

**Description:**

Verify by using admin or super user credentials that you can access the Users CRUD.

**Steps:**

1. Enter the homepage
2. Login with Admin or Super User credentials
3. Click on Users

**Test Validation:**

If you get the User CRUD on the home page with Admin or Super User credentials, the test passes.

Happy Path: Verify you can create a User in the User CRUD when logging in as an Admin or Super User

**Description:**

Verify by using admin or super user credentials that you can create a user in the Users CRUD.

**Steps:**

1. Enter the homepage
2. Login with Admin or Super User credentials
3. Click on Users
4. Click on Create
5. Enter valid information in all the fields. Select an English Level, the Password should have 8 characters, one upper case and one special character, and email should be in email format
6. Click Submit

**Test Validation:**

If the user gets created successfully, the test passes.

Happy Path: Verify you can see Users in the User CRUD when logging in as an Admin or Super User

**Description:**

Verify by using admin or super user credentials that you can access a list of the users in the Users CRUD.

**Steps:**

1. Enter the homepage
2. Login with Admin or Super User credentials
3. Click on Users
4. Click on List

**Test Validation:**

If you get the user list in the User CRUD on the home page with Admin or Super User credentials, the test passes.

*Happy Path:* Verify you can search a user in the User CRUD when logging in as an Admin or Super User

**Description:**

Verify by using admin or super user credentials that you can search a user in the Users CRUD.

**Steps:**

1. Enter the homepage
2. Login with Admin or Super User credentials
3. Click on Users
4. Enter the name of a user in the input of Search
5. Click Search

**Test Validation:**

If you get the information of a specific user in the User CRUD on the home page with Admin or Super User credentials, the test passes.

*Happy Path:* Verify you can delete a user in the User CRUD when logging in as an Admin or Super User

**Description:**

Verify by using admin or super user credentials that you can delete a user in the Users CRUD.

**Steps:**

1. Enter the homepage
2. Login with Admin or Super User credentials
3. Click on Users
4. Enter the name of a user in the input of Delete
5. Click Delete

**Test Validation:**

If you get the user deleted in the User CRUD on the home page with Admin or Super User credentials, the test passes.

Happy Path: Verify you can see the Teams CRUD when logging in as an Admin or Super User

**Description:**

Verify by using admin or super user credentials that you can access the Teams CRUD.

**Steps:**

1. Enter the homepage
2. Login with Admin or Super User credentials
3. Click on Teams

**Test Validation:**

If you get the Teams CRUD on the home page with Admin or Super User credentials, the test passes.

Happy Path: Verify you can create a team in the Teams CRUD when logging in as an Admin or Super User

**Description:**

Verify by using admin or super user credentials that you can create a team in the TeamsCRUD.

**Steps:**

7. Enter the homepage
8. Login with Admin or Super User credentials
9. Click on Teams
10. Click on Create
11. Enter valid information in all the fields. Enter a name and a description
12. Click Create

**Test Validation:**

If the team gets created successfully, the test passes.

Negative Scenario: Verify you can't create a user in the Users CRUD without a Name

**Description:**

Verify by using admin or super user credentials that you can't create a user in the Users CRUD without a Name.

**Steps:**

1. Enter the homepage
2. Login with Admin or Super User credentials
3. Click on Teams
4. Click on Create

5. Enter valid information in all the fields, but leave the Name field blank
6. Click Create

**Test Validation:**

If you get an error and does not create a user, the test passes.

Negative Scenario: Verify you can't create a user in the Users CRUD with an invalid Email

**Description:**

Verify by using admin or super user credentials that you can't create a user in the Users CRUD without a valid email.

**Steps:**

1. Enter the homepage
2. Login with Admin or Super User credentials
3. Click on Users
4. Click on Create
5. Enter valid information in all the fields, but in Email put an invalid email, such a random string of numbers
6. Click Create

**Test Validation:**

If you get an error and does not create a user, the test passes.

Negative Scenario: Verify you can't create a user in the Users CRUD without a password

**Description:**

Verify by using admin or super user credentials that you can't create a user in the Users CRUD without a password.

**Steps:**

1. Enter the homepage
2. Login with Admin or Super User credentials
3. Click on Users
4. Click on Create
5. Enter valid information in all the fields, but leave the Password and Confirm Password field blank
6. Click Create

**Test Validation:**

If you get an error and does not create a user, the test passes.

Negative Scenario: Verify you can't create a user in the Users CRUD without a CV

**Description:**

Verify by using admin or super user credentials that you can't create a user in the Users CRUD without a CV.

**Steps:**

1. Enter the homepage
2. Login with Admin or Super User credentials
3. Click on Users
4. Click on Create
5. Enter valid information in all the fields, but leave the CV field blank
6. Click Create

**Test Validation:**

If you get an error and does not create a user, the test passes.

Negative Scenario: Verify you can't create a user in the Users CRUD without Experience

**Description:**

Verify by using admin or super user credentials that you can't create a user in the Users CRUD without experience.

**Steps:**

1. Enter the homepage
2. Login with Admin or Super User credentials
3. Click on Users
4. Click on Create
5. Enter valid information in all the fields, but leave the Experience field blank
6. Click Create

**Test Validation:**

If you get an error and does not create a user, the test passes.

Negative Scenario: Verify you can't create a user in the Users CRUD without an English level

**Description:**

Verify by using admin or super user credentials that you can't create a user in the Users CRUD without English level.

**Steps:**

1. Enter the homepage
2. Login with Admin or Super User credentials



3. Click on Users
4. Click on Create
5. Enter valid information in all the fields, but leave the English Level dropdown field blank
6. Click Create

**Test Validation:**

If you get an error and does not create a user, the test passes.

Negative Scenario: Verify you can't create a user in the Users CRUD with a different Password in the confirmation

**Description:**

Verify by using admin or super user credentials that you can't create a user in the Users CRUD with a different Password in the confirmation.

**Steps:**

1. Enter the homepage
2. Login with Admin or Super User credentials
3. Click on Users
4. Click on Create
5. Enter valid information in all the fields, but put a different password in the Confirm Password
6. Click Create

**Test Validation:**

If you get an error and does not create a user, the test passes.

Happy Path: Verify you can see teams in the Teams CRUD when logging in as an Admin or Super User

**Description:**

Verify by using admin or super user credentials that you can access a list of the teams in the Teams CRUD.

**Steps:**

5. Enter the homepage
6. Login with Admin or Super User credentials
7. Click on Teams
8. Click on List

**Test Validation:**

If you get the teams list in the Teams CRUD on the home page with Admin or Super User credentials, the test passes.

Happy Path: Verify you can search a team in the Teams CRUD when logging in as an Admin or Super User

**Description:**

Verify by using admin or super user credentials that you can search a team in the Teams CRUD.

**Steps:**

6. Enter the homepage
7. Login with Admin or Super User credentials
8. Click on Teams
9. Enter the name of a team in the input of Search
10. Click Search

**Test Validation:**

If you get the information of a specific team in the Teams CRUD on the home page with Admin or Super User credentials, the test passes.

Happy Path: Verify you can delete a team in the Teams CRUD when logging in as an Admin or Super User

**Description:**

Verify by using admin or super user credentials that you can delete a team in the Teams CRUD.

**Steps:**

6. Enter the homepage
7. Login with Admin or Super User credentials
8. Click on Teams
9. Enter the name of a team in the input of Delete
10. Click Delete

**Test Validation:**

If you get the team deleted in the Teams CRUD on the home page with Admin or Super User credentials, the test passes.

Negative Scenario: Verify you can't create a team in the Teams CRUD without a Name

**Description:**

Verify by using admin or super user credentials that you can't create a team in the Teams CRUD without a Name.

**Steps:**

1. Enter the homepage
2. Login with Admin or Super User credentials
3. Click on Teams
4. Click on Create
5. Enter valid information in all the fields, but leave the Name field blank
6. Click Create

**Test Validation:**

If you get an error and does not create a user, the test passes.

Negative Scenario: Verify you can't create a team in the Teams CRUD without a Description

**Description:**

Verify by using admin or super user credentials that you can't create a team in the Teams CRUD without a Description.

**Steps:**

1. Enter the homepage
2. Login with Admin or Super User credentials
3. Click on Teams
4. Click on Create
5. Enter valid information in all the fields, but leave the Name field blank
6. Click Create

**Test Validation:**

If you get an error and does not create a user, the test passes.

Happy Path: Verify you can see the Accounts CRUD when logging in as an Admin or Super User

**Description:**

Verify by using admin or super user credentials that you can access the Accounts CRUD.

**Steps:**

1. Enter the homepage
2. Login with Admin or Super User credentials
3. Click on Accounts

**Test Validation:**

If you get the Accounts CRUD on the home page with Admin or Super User credentials, the test passes.

Happy Path: Verify you can create an account in the Accounts CRUD when logging in as an Admin or Super User

**Description:**

Verify by using admin or super user credentials that you can create an account in the Accounts CRUD.

**Steps:**

7. Enter the homepage
8. Login with Admin or Super User credentials
9. Click on Accounts
10. Click on Create
11. Enter valid information in all the fields. Enter a name, a customer name, an operational manager and an existing team
12. Click Create

**Test Validation:**

If the account gets created successfully, the test passes.

Negative Scenario: Verify you can't create an account in the Accounts CRUD without a Name

**Description:**

Verify by using admin or super user credentials that you can't create an account in the Accounts CRUD without a Name.

**Steps:**

1. Enter the homepage
2. Login with Admin or Super User credentials
3. Click on Accounts
4. Click on Create
5. Enter valid information in all the fields, but leave the Name field blank
6. Click Create

**Test Validation:**

If you get an error and does not create a user, the test passes.

Negative Scenario: Verify you can't create an account in the Accounts CRUD without a Customer

**Description:**

Verify by using admin or super user credentials that you can't create an account in the Accounts CRUD without a Name.

**Steps:**

1. Enter the homepage
2. Login with Admin or Super User credentials
3. Click on Accounts
4. Click on Create
5. Enter valid information in all the fields, but leave the Customer name field blank
6. Click Create

**Test Validation:**

If you get an error and does not create a user, the test passes.

Negative Scenario: Verify you can't create an account in the Accounts CRUD without a Manager

**Description:**

Verify by using admin or super user credentials that you can't create an account in the Accounts CRUD without a Manager.

**Steps:**

1. Enter the homepage
2. Login with Admin or Super User credentials
3. Click on Accounts
4. Click on Create
5. Enter valid information in all the fields, but leave the Operational Manager field blank
6. Click Create

**Test Validation:**

If you get an error and does not create a user, the test passes.

Negative Scenario: Verify you can't create an account in the Accounts CRUD without a Team

**Description:**

Verify by using admin or super user credentials that you can't create an account in the Accounts CRUD without a Name.

**Steps:**

1. Enter the homepage
2. Login with Admin or Super User credentials
3. Click on Accounts
4. Click on Create
5. Enter valid information in all the fields, but leave the Name field blank
6. Click Create

**Test Validation:**

If you get an error and does not create a user, the test passes.

*Happy Path:* Verify you can see accounts in the Accounts CRUD when logging in as an Admin or Super User

**Description:**

Verify by using admin or super user credentials that you can access a list of the accounts in the Accounts CRUD.

**Steps:**

9. Enter the homepage
10. Login with Admin or Super User credentials
11. Click on Accounts
12. Click on List

**Test Validation:**

If you get the accounts list in the Accounts CRUD on the home page with Admin or Super User credentials, the test passes.

*Happy Path:* Verify you can search an account in the Accounts CRUD when logging in as an Admin or Super User

**Description:**

Verify by using admin or super user credentials that you can search an account in the Accounts CRUD.

**Steps:**

11. Enter the homepage
12. Login with Admin or Super User credentials
13. Click on Accounts
14. Enter the name of an account in the input of Search
15. Click Search

**Test Validation:**

If you get the information of a specific account in the Accounts CRUD on the home page with Admin or Super User credentials, the test passes.

*Happy Path:* Verify you can delete an account in the Accounts CRUD when logging in as an Admin or Super User

**Description:**

Verify by using admin or super user credentials that you can delete an account in the Accounts CRUD.

**Steps:**

11. Enter the homepage
12. Login with Admin or Super User credentials
13. Click on Accounts
14. Enter the name of an account in the input of Delete
15. Click Delete

**Test Validation:**

If you get the account deleted in the Accounts CRUD on the home page with Admin or Super User credentials, the test passes.

*Happy Path:* Verify you can see the Operations CRUD when logging in as an Admin or Super User

**Description:**

Verify by using admin or super user credentials that you can access the Operations CRUD.

**Steps:**

1. Enter the homepage
2. Login with Admin or Super User credentials
3. Click on Operations

**Test Validation:**

If you get the Operations CRUD on the home page with Admin or Super User credentials, the test passes.

*Happy Path:* Verify you can View History in the Operations CRUD when logging in as an Admin or Super User

**Description:**

Verify by using admin or super user credentials that you can access the Operations CRUD.

**Steps:**

1. Enter the homepage
2. Login with Admin or Super User credentials
3. Click on Operations

**Test Validation:**

If you get the Operations CRUD on the home page with Admin or Super User credentials, the test passes.

*Happy Path:* Verify you can View History by dates in the Operations CRUD when logging in as an Admin or Super User

**Description:**

Verify by using admin or super user credentials that you can access the Operations CRUD.

**Steps:**

1. Enter the homepage
2. Login with Admin or Super User credentials
3. Click on Operations

**Test Validation:**

If you get the Operations CRUD on the home page with Admin or Super User credentials, the test passes.

*Happy Path:* Verify you can View History by user in the Operations CRUD when logging in as an Admin or Super User

**Description:**

Verify by using admin or super user credentials that you can access the Operations CRUD.

**Steps:**

1. Enter the homepage
2. Login with Admin or Super User credentials
3. Click on Operations

**Test Validation:**

If you get the Operations CRUD on the home page with Admin or Super User credentials, the test passes.

*Happy Path:* Verify you can View History by team in the Operations CRUD when logging in as an Admin or Super User

**Description:**

Verify by using admin or super user credentials that you can access the Operations CRUD.

**Steps:**

1. Enter the homepage
2. Login with Admin or Super User credentials
3. Click on Operations

**Test Validation:**

If you get the Operations CRUD on the home page with Admin or Super User credentials, the test passes.



Happy Path: Verify you can move users between teams in the Operations CRUD when logging in as an Admin or Super User

**Description:**

Verify by using admin or super user credentials that you can access the Operations CRUD.

**Steps:**

1. Enter the homepage
2. Login with Admin or Super User credentials
3. Click on Operations

**Test Validation:**

If you get the Operations CRUD on the home page with Admin or Super User credentials, the test passes.

### 3.3. Bugs and Improvements

You are able to Search by ID but you get an error in console

#### Expected Result:

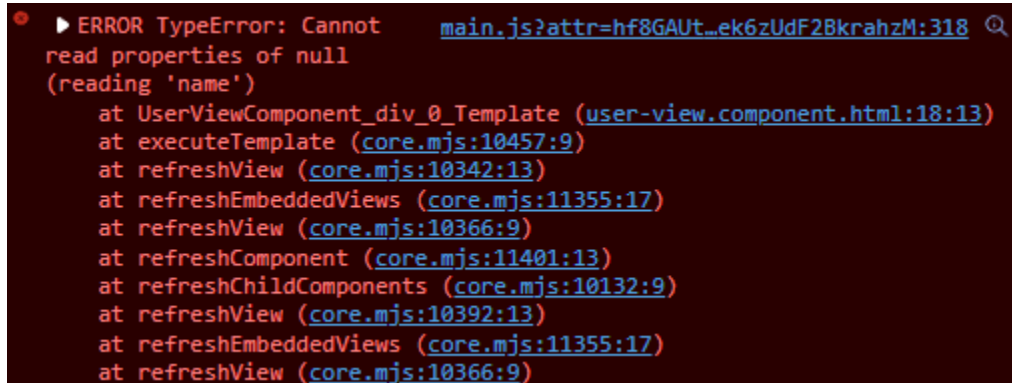
You only should be able to search by name.

#### Actual Result:

When searching a user, a team or an account by its ID, you get the correct result, but you get an error in the console.

#### Steps:

1. Open the Home page
2. Log in with Admin or Super User credentials
3. Go to Users, Teams or Accounts
4. Fill a User ID, Teams ID or Account ID in the input of Search
5. Click Search
6. Verify that the returned information is correct
7. Verify that you get an error in the Console



```
ERROR TypeError: Cannot read properties of null (reading 'name')
    at UserViewComponent_div_0_Template (user-view.component.html:18:13)
    at executeTemplate (core.mjs:10457:9)
    at refreshView (core.mjs:10342:13)
    at refreshEmbeddedViews (core.mjs:11355:17)
    at refreshView (core.mjs:10366:9)
    at refreshComponent (core.mjs:11401:13)
    at refreshChildComponents (core.mjs:10132:9)
    at refreshView (core.mjs:10392:13)
    at refreshEmbeddedViews (core.mjs:11355:17)
    at refreshView (core.mjs:10366:9)
```

You are not able to Search by Name

#### Expected Result:

You only should be able to search by name.

#### Actual Result:

When searching a user, a team or an account by its ID, you get the correct result, but you get an error in the console.

#### Steps:

1. Open the Home page
2. Log in with Admin or Super User credentials
3. Go to Users, Teams or Accounts
4. Fill a User ID, Teams ID or Account ID in the input of Search

5. Click Search
6. Verify that you get this error in console:

```
{
  "headers": {
    "normalizedNames": {},
    "lazyUpdate": null
  },
  "status": 400,
  "statusText": "OK",
  "url": "https://localhost:44309/api/Users/Daniel",
  "ok": false,
  "name": "HttpErrorResponse",
  "message": "Http failure response for https://localhost:44309/api/Users/Daniel: 400 OK",
  "error": {
    "type": "https://tools.ietf.org/html/rfc7231#section-6.5.1",
    "title": "One or more validation errors occurred.",
    "status": 400,
    "traceId": "00-92ad11711952bf9feb6bbc41c590c16c-2f3e599d070737ee-00",
    "errors": {
      "id": [
        "The value 'Daniel' is not valid."
      ]
    }
  }
}
```

You get to a blank page if you search non-existence ID

**Expected Result:**

You should not be able to search a non-existence row in the DB.

**Actual Result:**

When searching a user, a team or an account by its ID, and that ID does not exist in the DB, you get to a blank page. You don't get an error, and console returns null.

**Steps:**

- 1. Open the Home page
- 2. Log in with Admin or Super User credentials
- 3. Go to Users, Teams or Accounts
- 4. Fill a non-existence User ID, Teams ID or Account ID in the input of Search
- 5. Click Search
- 6. Verify that you get to a blank page

View History, View History by User, and View History by Team does not store the dates

**Expected Result:**

You should be able to look at the date when a movement was made in View History, View History by User, and View History by Team.

**Actual Result:**

When searching in View History, View History by User, and View History by Team, you don't get to see any date stored.

**Steps:**

- 1. Open the Home page
- 2. Log in with Admin or Super User credentials
- 3. Go to Operations
- 4. Select View History, View History by User or View History by Team page and verify no date is stored in the table:

**Operations**

SuperAdmin▼search

| User       | Team left | Team Joined | Date |
|------------|-----------|-------------|------|
| SuperAdmin | None      | Group1      |      |

cancel

View History by Dates does not change

### Expected Result:

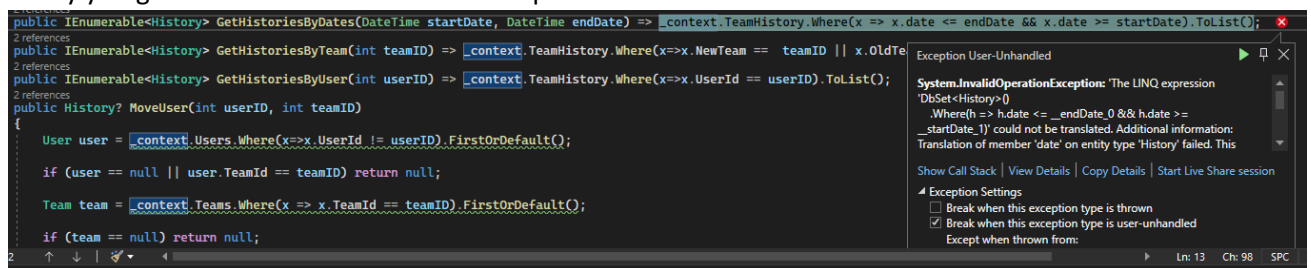
You should be able to search History by filtering dates.

### Actual Result:

When searching an operation's history by dates, the internal process stops and you're not able to continue with anything until you reload the whole backend.

### Steps:

1. Open the Home page
2. Log in with Admin or Super User credentials
3. Go to Operations
4. Select View History by Dates
5. Select a starting date and an end date
6. Click Search
7. Verify you get this error in the backend compiler:



View History by User randomly fills User as "None"

### Expected Result:

You should be able the values on the table all the times.

### Actual Result:

When searching in View History by User, going back and forth, gets you None as value in the names of the users.

### Steps:

1. Open the Home page
2. Log in with Admin or Super User credentials
3. Go to Operations
4. Go to View History by User
5. Search a specific user by selecting it in the dropdown menu and clicking search
6. Cancel

- Repeat step 5 and 6 until on Step 5 you see the table with “None” as names as shown in the screenshot:

#### Operations

| User | Team left | Team Joined | Date |
|------|-----------|-------------|------|
| None | None      | Group1      |      |
| None | Group1    | Team A      |      |
| None | None      | AAAAA       |      |

cancel

Credentials still logged in after removing user from DB

#### Expected Result:

When finishing the process of the backend and starting it back again, credentials should be logged off.

#### Actual Result:

When removing all users in the database and a credential was already logged in, if restarting the backend, the credential is still logged in, although the user was removed.

#### Steps:

- Open the Home page
- Log in with Admin or Super User credentials
- Delete the user in the DB
- Restart the backend
- Open the Home page again

Delete CRUD does not exist

#### Expected Result:

You should be able to delete users, teams and accounts in the CRUD.

#### Actual Result:

When entering the CRUDs, there is no Delete/Remove option.

#### Steps:

- Open the Home page

Create Account API response returns a null Team value that is not needed

**Expected Result:**

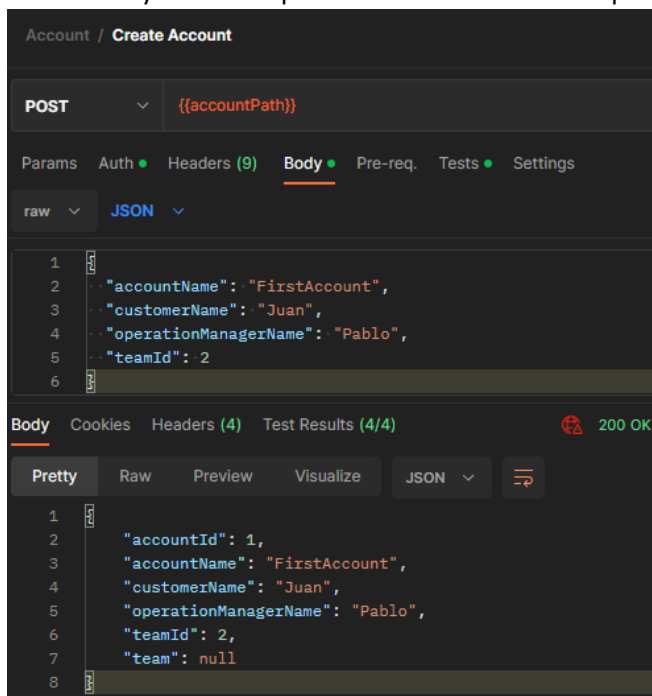
API Response from Account creation should only include teamId.

**Actual Result:**

API Response from Account creation returned the team name as null, which value should not be returned at all.

**Steps:**

1. Open Postman
2. Open Create Account request
3. Fill the Body's JSON requirements and send the request



Cancel button says cancel on Lists

**Expected Result:**

Cancel button should say Cancel.

**Actual Result:**

Cancel button says cancel.

**Steps:**

1. Open the Home page
2. Log in with admin or super user credentials
3. Open any CRUD

4. Click on List
5. Check the Cancel button

