**HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY**

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY



**PROJECT 1 REPORT**

**Managing Microsoft Teams Services and Syncing Data with AirTable using Microsoft Graph API and AirTable API**

Nguyen Thien Sang 20214972

Pham Lam Phu 20214969

**Hanoi, 2023**

Table of Contents

[1. Task assignment 2](#_Toc141297846)

[2. Introduction 3](#_Toc141297847)

[3. Annotation 3](#_Toc141297848)

[4. Methodology 5](#_Toc141297849)

[5. Analysis and Design 6](#_Toc141297850)

[5.1 Package dependency analysis 6](#_Toc141297851)

[5.2 Class diagram analysis 6](#_Toc141297852)

[5.2.1. *template.colorUtil* 6](#_Toc141297853)

[5.2.2. *template.jsonUtil* 6](#_Toc141297854)

[5.2.3. *template.persistence.dto* 7](#_Toc141297855)

[5.2.4. *template.service* 8](#_Toc141297856)

[5.2.5. *template.service.airtable* 13](#_Toc141297857)

[5.2.6. *template.accessInfo* 19](#_Toc141297858)

[5.2.7. *template.team\_config* 20](#_Toc141297859)

[5.2.8. *CLI.java* 21](#_Toc141297860)

[5.3 Activity diagram 21](#_Toc141297861)

[6. Demo and Evaluation 35](#_Toc141297862)

[6.1. Demo 35](#_Toc141297863)

[6.2. Evaluation 47](#_Toc141297864)

[6.2.1. Check by SonarQube (static code) 47](#_Toc141297865)

[6.2.2. Dynamic code analysis 48](#_Toc141297866)

[6.2.3. Secure coding 48](#_Toc141297867)

[7. Configuration guide 48](#_Toc141297868)

# **1. Task assignment**

|  |  |  |
| --- | --- | --- |
| **Student name** | **Student id** | **Task** |
| **Phạm Lâm Phú** | **20214969** | -Build the base structure of the project  -Develope some user management function using Microsoft Graph API  -Build task schedule script to run program automatically  -Fix bug,handle exception  -Draw acitivity diagram  -Write report |
| **Nguyễn Thiện Sang** | **20214972** | -Develop sync functions using AirTable API  -Develop some functions about Microsoft Teams: create channel, create link to Team, list users in a organization, list all channels, list existed Team and add new field (joinedTeams) to AirTable.  -Build CLI (Command-line Interface) -Create ClassDiagram  -Fix bug, handle exception  -Write report |

# **2. Introduction**

Synchronizing data with AirTable is a common task in modern software development. Microsoft Team is a popular messaging and collaboration platform used by many organizations, while AirTable is a cloud-based database management system that allows users to store, organize, and analyze data in a user-friendly interface.

The program allows administrators to manage users, channels, and other settings for their Microsoft Team services. It also provides the ability to synchronize data with AirTable, ensuring that all relevant information is up-to-date and easily accessible. This synchronization can be achieved through a variety of methods, including API calls and webhooks.

# **3. Annotation**

In package ***template.persistence.dto***

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | ***Parameter*** | ***Type*** | ***Description*** |
| ***User*** | *displayName* | *String* | The name to display in the address book for the user. |
| *userPrincipalName* | *String* | The user principal name (someuser@contoso.com). It's an Internet-style login name for the user based on the Internet standard RFC 822. By convention, this should map to the user's email name. The general format is alias@domain, where domain must be present in the tenant's collection of verified domains. The verified domains for the tenant can be accessed from the verifiedDomains property of organization.  NOTE: This property cannot contain accent characters. Only the following characters are allowed A - Z, a - z, 0 - 9, ' . - \_ ! # ^ ~. For the complete list of allowed characters, see username policies. |
| *mailNickname* | *String* | The mail alias for the user. |
| *password* | *String* | The password profile for the user. |
| ***Group*** | *displayName* | *String* | The name to display in the address book for the group. Maximum length: 256 characters. Required. |
| *mailNickname* | *String* | The mail alias for the group, unique for Microsoft 365 groups in the organization. Maximum length is 64 characters. This property can contain only characters in the ASCII character set 0 - 127 except the following: @ () \ [] " ; : <> , SPACE. Required. |
| *description* | *String* | The description for the group |

Reference: <https://learn.microsoft.com/en-us/graph/api/user-post-users?view=graph-rest-1.0&tabs=http>

<https://learn.microsoft.com/en-us/graph/api/group-post-groups?view=graph-rest-1.0&tabs=http>

# **4.** **Methodology**

In this project, we used Java as the programming language. To program functions and utilities for the application, we also need assistance from both JDK 20,Maven and other external libraries including:

* Org.apache.httpcomponents : use to send request to microsoft azure to use api
* Org.json : use to read json file
* com.google.gson : Read the HTTP response body
* org.apache.poi : write data from table on airtable to csv file
* java.util.logging : write log file to record activity log
* java.io : read file and handle exception
* java.net: use to send HTTP request
* org.fusesource.jansi: print text with defined color

Otherwise , we use org.codehaus.mojo plugin to define maven project to help task schedule work

# **5. Analysis and Design**

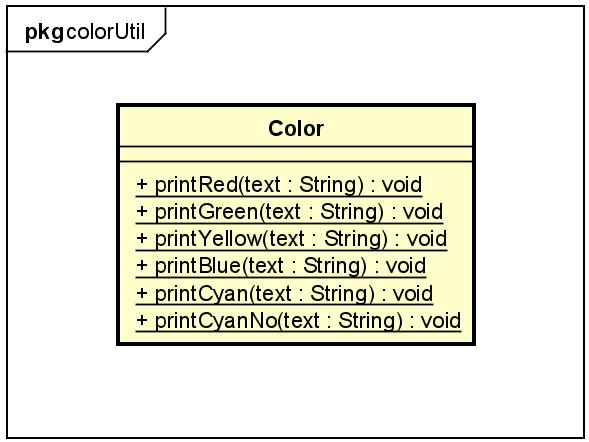
## **5.1 Package dependency analysis**

A diagram of a computer file

Description automatically generated

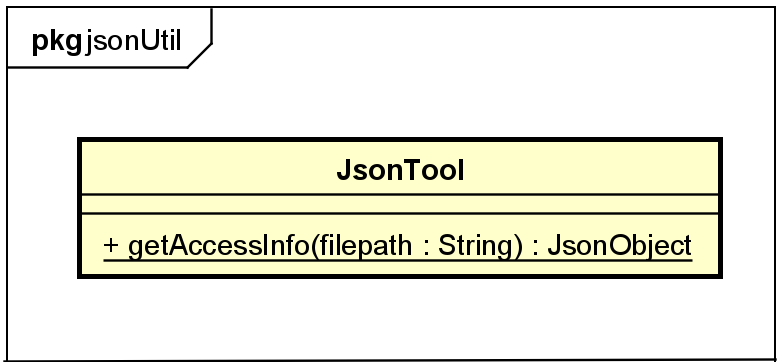
## **5.2 Class diagram analysis**

### **5.2.1. *template.colorUtil***



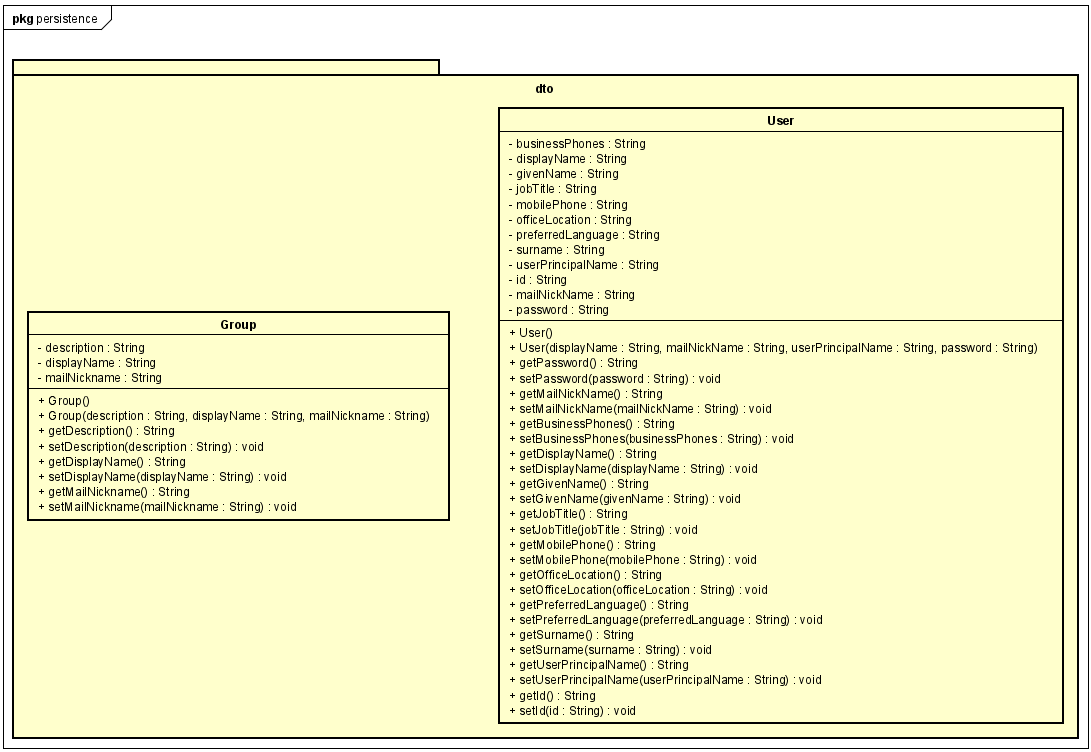
* *Color*: used for printing colored text to the terminal
  + *printRed(text : String)*: Print red text
  + *printGreen(text : String)*: Print green text
  + *printYellow(text : String)*: Print yellow text
  + *printYellowNo(text : String)*: Print yellow text without new line
  + *printBlue(text : String)*: print blue text
  + *printCyan(text : String)*: print cyan text
  + *printCyanNo(text : String)*: print cyan text without new line

### **5.2.2. *template.jsonUtil***



* *JsonTool*: used for converting data in .json file to JsonObject (com.google.gson.JsonObject)
  + *getAccessInfo(filepath : String)*: Read a JSON file and convert its contents into a JsonObject, return it as JsonObject (com.google.gson.JsonObject)

### **5.2.3. *template.persistence.dto***

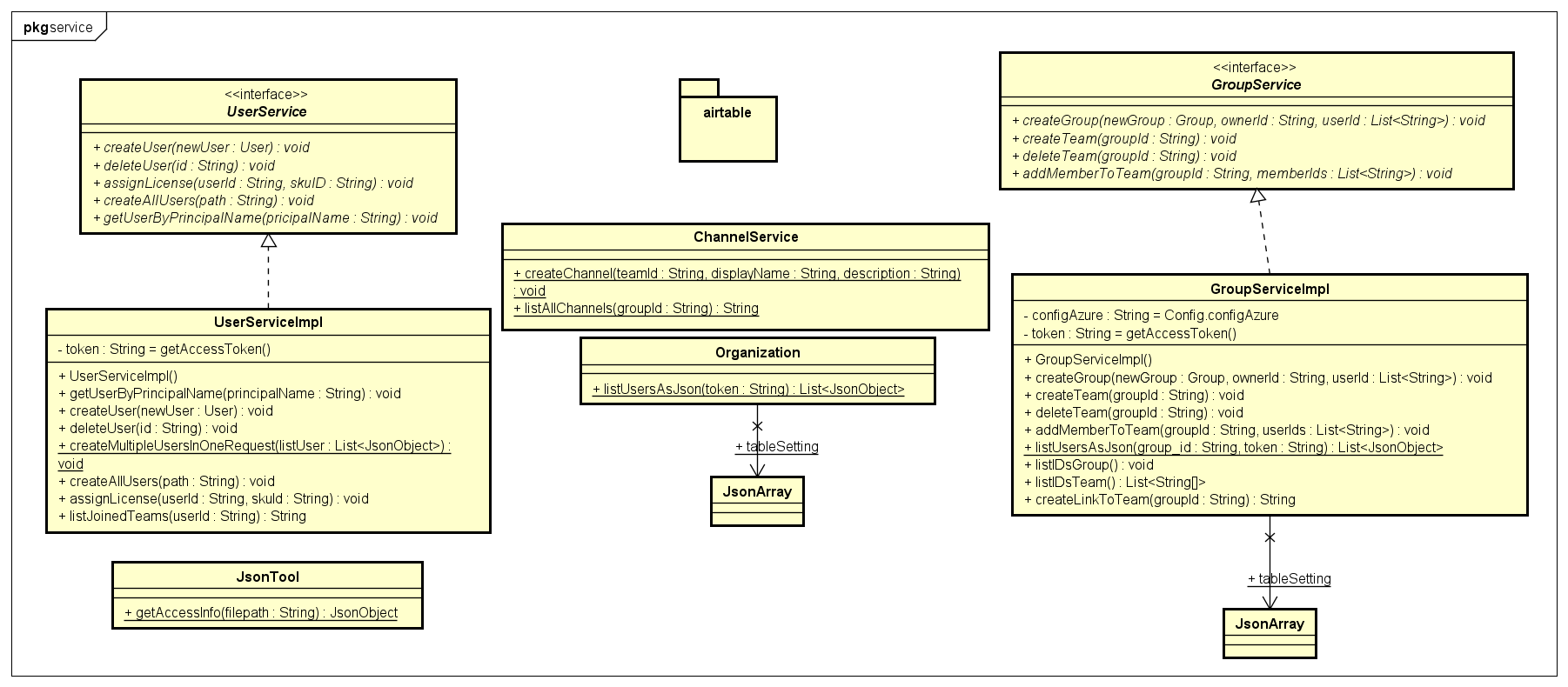


* *Group:*
  + *Group(String description ,String displayName ,String mailNickname) :* Constructor, include a description , a displayName and a mailNickname to get the some basic attributes of group
  + *Group() :* null constructor , use for some special case
  + *Getter and Setter :* use to access and modify the values of an group's attributes

(these attribute also use for teams)

* *User:*
  + *User(String displayName, String mailNickName, String userPrincipalName, String password) :* Constructor, include a displayName , a mailNickname , a userPrincipalName and a password to get the some basic attributes of user
  + *User() :* null constructor , use for some special case
  + *Getter and Setter :* use to access and modify the values of an group's attributes

### **5.2.4. *template.service***



* *UserService :*
  + *createUser(User : newUser)*: This function allows the user to create a new user by providing
    - Display name: displayName
    - Abbreviated name for email: mailNickname
    - Main Username: userPrincipalName
    - Password: password

(note :

+mailNickName and userPrincipalName cannot be duplicated

+ Password must contain 8 characters, including at least 1 uppercase letter and 1 special character)

(reference : [https://learn.microsoft.com/en-us/graph/api/user-post-users?view=graph-rest-1.0&tabs=http )](https://learn.microsoft.com/en-us/graph/api/user-post-users?view=graph-rest-1.0&tabs=http)

* + *createAllUsers (String : path)* : This function allows users to create multiple users at the same time by providing a CSV file containing user information. The result after creation will be displayed in the file “batch\_request.log”

(reference :

<https://learn.microsoft.com/en-us/graph/api/user-post-users?view=graph-rest-1.0&tabs=http> ;

<https://stackoverflow.com/questions/52994923/microsoft-graph-api-how-to-create-multiple-users-at-once>)

* + *deleteUser (String : id)*: This function allows the user to delete an existing user from the service by providing
    - User ID :userID

(note: userID must exist or the request will return an error)

(reference :<https://learn.microsoft.com/en-us/graph/api/user-delete?view=graph-rest-1.0&tabs=http>)

* + *getUserByPrincipalName (String : pricipalName)* : This function allows the user to get an existing user from the service by providing
    - The primary name of the user: *userPrincipalName*

(note: *userPrincipalName* must exist)

(reference [https://learn.microsoft.com/en-us/](https://learn.microsoft.com/en-us/graph/api/user-get?view=graph-rest-1.0&tabs=http) graph/api/user-get?view=graph-rest- 1.0&tabs=http)

* + *getAllUsers()* : This function allows users to get all existing users in the organization from the service

(reference : [https://learn.microsoft.com/en-us](https://learn.microsoft.com/en-us/graph/api/user-list?view=graph-rest-1.0&tabs=http) /graph/api/user-list?view=graph-rest-1.0&tabs=http)

* + *assignLicense(String userId)*: This function allows a user to assign a license to an existing user by providing:
    - User ID: *userId*

(note: *userId* must exist)

(reference:<https://learn.microsoft.com/en-us/graph/api/user-assignlicense?view=graph-rest-1.0&tabs=http>)

* + *listJoinedTeams(userId : String)*: Used for listing all Team that userId joined as String  
    Reference: <https://learn.microsoft.com/en-us/graph/api/user-list-joinedteams?view=graph-rest-1.0&tabs=http>
* *GroupService:*
  + *createGroup(Group : newGroup, String : ownerId , List<String> : userId)* : This function allows the user to create a new group by providing :
    - Display name: *displayName*
    - Description: *description*
    - Abbreviated name for mail: *mailNickName*
    - Owner ID: *ownerId*
    - User ID: *userId* (multiple users can be entered)

(note: *ownerId* and *userId* must exist)

In addition, a Team is created from the group using the createTeam method.

(reference :<https://learn.microsoft.com/en-us/graph/api/group-post-groups?view=graph-rest-1.0&tabs=http>)

* + *createTeam(String groupId)*: This function allows users to create a team for an existing team by providing
    - Group ID: *groupID*

(note: *groupId* must exist)

(reference : [https://learn.microsoft.com/en-us](https://learn.microsoft.com/en-us/graph/api/team-put-teams?view=graph-rest-1.0&tabs=http) /graph/api/team-put-teams?view=graph-rest-1. 0&tabs=http)

* + *Delete Team (String : groupId)* : This function allows the user to delete a team for an existing team by providing :
    - Group ID: *groupId*

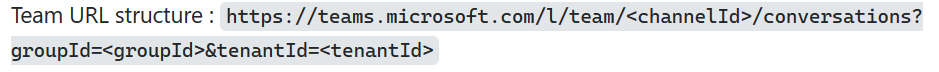
(note: *groupId* must exist)

(reference :<https://learn.microsoft.com/en-us/graph/api/group-delete?view=graph-rest-1.0&tabs=http>)

* + *addMemberToTeam(String groupId,List<String> memberIds)* : This function allows users to add one or more users to an existing team by providing :
    - Group ID: *groupId*
    - User ID: *usersId*

(note: *groupId* must exist)

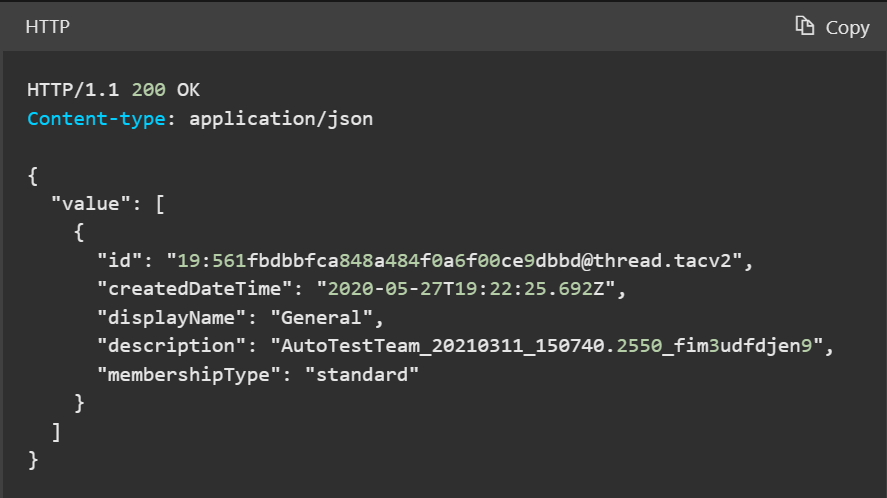
(reference :<https://learn.microsoft.com/en-us/graph/api/group-post-members?view=graph-rest-1.0&tabs=http>)

* + *GroupServiceImpl*:
* *listIDsGroup()*: method, print all groups id in an organization.  
  Reference: <https://learn.microsoft.com/en-us/graph/api/group-list?view=graph-rest-1.0&tabs=http>
* *listIDsTeam()*: method, return the *List<String[]>*. Each array of *String* in List will contain *id* (*groupId*) and *name* (*displayName*) of groups that contain Team in an organization.  
  Reference: <https://learn.microsoft.com/en-us/graph/teams-list-all-teams>
* *createLinkToTeam(groupId : String)*: class method, return link (invitation) to Team that has id = *groupId*.  
    
  Reference: <https://stackoverflow.com/questions/61433019/how-can-we-programmatically-generate-ms-teams-code>
  + *ChannelService:*
  + *createChannel(teamId : String, displayName : String, description : String)*: This method allows the user to create a new channel for an existing group by providing:
    - Group ID: groupID
    - Display name: displayName
    - Described description

(note: groupID must exist)

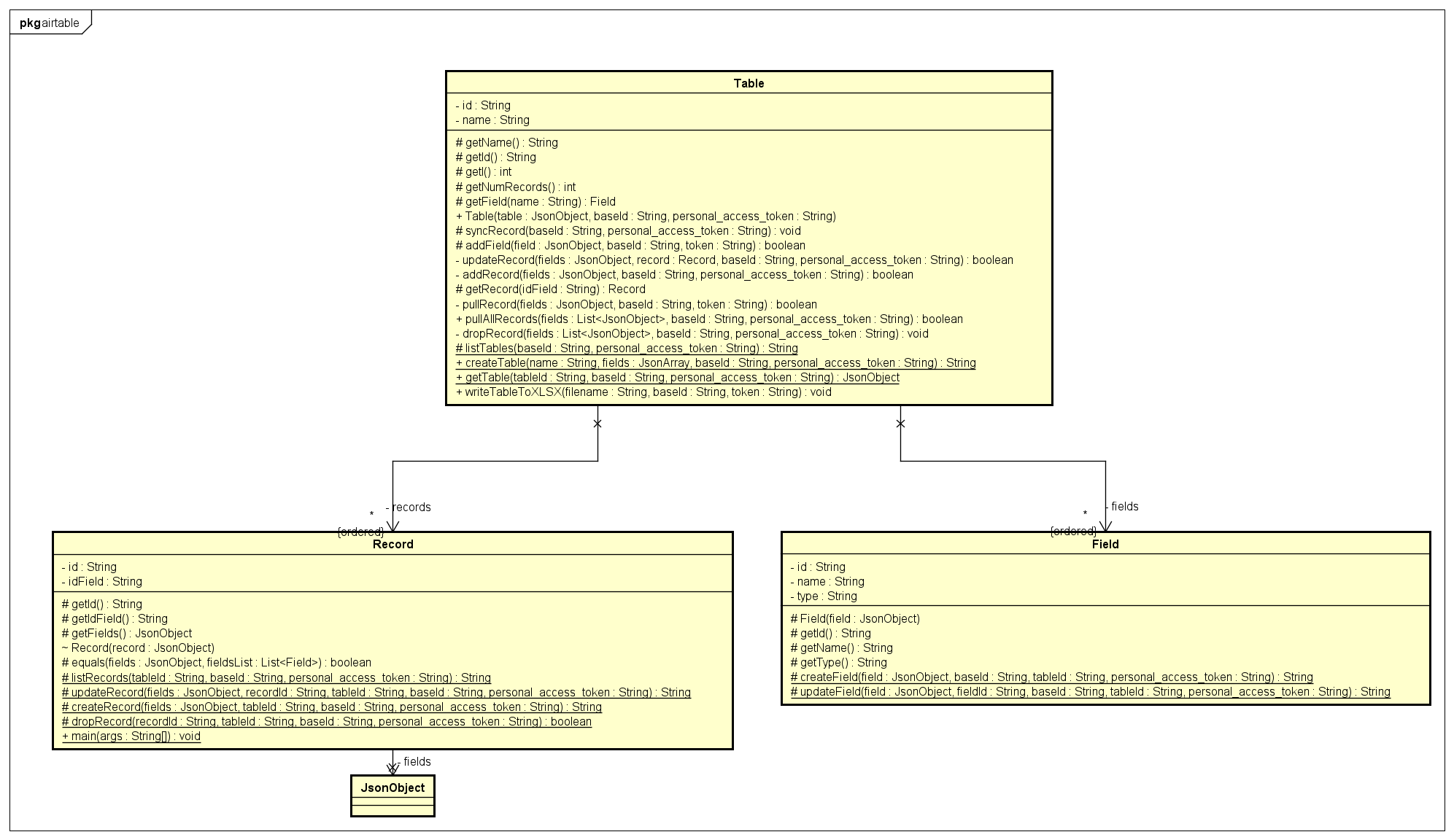
(reference : [https://learn.microsoft.com/en-us/graph/api/channel-](https://learn.microsoft.com/en-us/graph/api/channel-post?view=graph-rest-1.0&tabs=http) post?view=graph-rest-1.0&tabs=http)

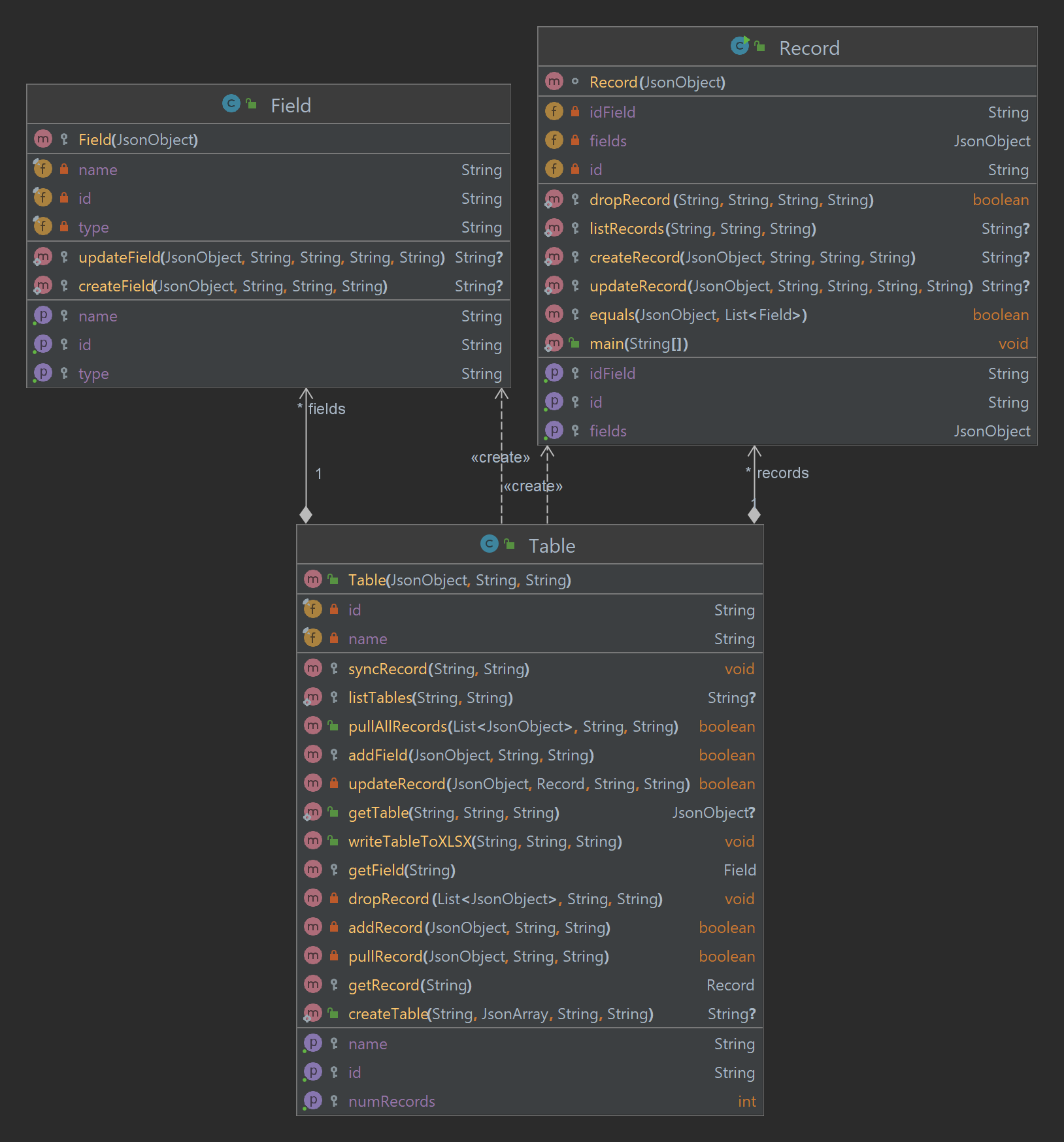
* + *listAllChannels(groupId : String)*: method, using HTTP Request to get the information of all channels in Team that has id = *groupId*. Return the response body of request as String, return *null* if error occurs.   
    The following is the response



Reference: [https://learn.microsoft.com/en-us/graph/api/channel-list?v](https://learn.microsoft.com/en-us/graph/api/channel-list?view=graph-rest-1.0&tabs=http) iew=graph-rest-1.0&tabs=http

### **5.2.5. *template.service.airtable***

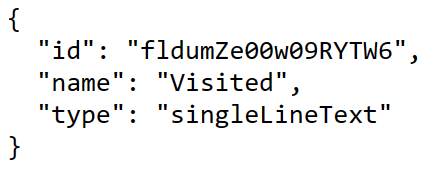




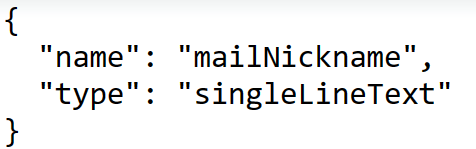
* *Field*:
  + *name*: name of the field
  + *id*: id of the field
  + *type*: type of the field

Reference: <https://airtable.com/developers/web/api/field-model>)

* + *Field(field : JsonObject)*: constructor, field(JsonObject) has the form of



* + *createField(field : JsonObject, baseId : String, tableId : String, personal\_access\_token : String)*: method to creates a new column using HTTP request, will take your input including:
    - *field(JsonObject)*, make sure this argument has the form



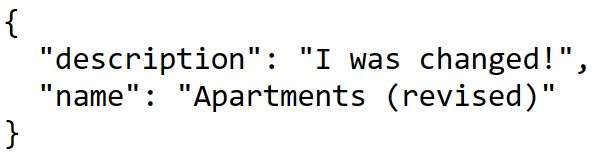
* *baseId(String)*: your baseId in AirTable
* *tableId(String)*: your tableId in AirTable
* *personal\_access\_token(String)*: your personal access token in AirTable

The response body of HTTP request will contain: *id, name, type* of a field

This method will return the response body as String

Reference: <https://airtable.com/developers/web/api/create-field>

* + *updateField(field : JsonObject, fieldId : String, baseId : String, tableId : String, personal\_access\_token : String)*: method to update the name and/or description of a field using HTTP request, will take your input including:
    - field(JsonObject), make sure this argument has the form of



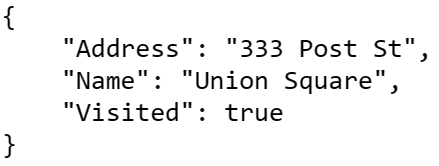
* *fieldId(String)*: id of the field in table
* *baseId(String)*: your baseId in AirTable
* *tableId(String)*: your tableId in AirTable
* *personal\_access\_token(String)*: your personal access token in AirTable

The response body will contain: *description, id, name, type* of updated field

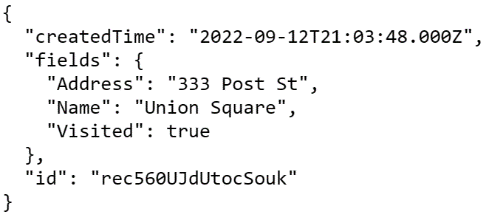
This method will return the response body as String

Reference: <https://airtable.com/developers/web/api/update-field>

* *Record*:
  + *idField(String)*: the id of user in team/group when do with MS Teams/Group or the userId of user when do with Organization
  + *fields(JsonObject)*: contain key (name of field) and value of a record, has the form of



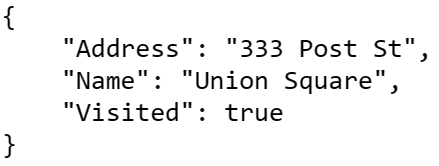
* + *id(String)*: id of a record in table
  + *Record(record : JsonObject)*: constructor, *record(JsonObject)* has the form of



* + *dropRecord(recordId : String, tableId : String, baseId : String, personal\_access\_token : String)*: method, used for dropping/deleting a single record that has id = *recordId* in a table using HTTP request. This method will return *true* if deleted successfully else *false*. Reference: <https://airtable.com/developers/web/api/delete-record>
  + *listRecords(tableId : String, baseId : String, personal\_access\_token : String)*: method, used for listing records in a table (using HTTP request).
    - *tableId*: your tableID (Note that table names and table ids can be used interchangeably. We recommend using table IDs so you don't need to modify your API request when your table name changes)

Reference: <https://airtable.com/developers/web/api/list-records>

* + *createRecord(fields : JsonObject, tableId : String, baseId : String, personal\_access\_token : String)*: method, used for creating new record (use HTTP request)
    - *fields(JsonObject)*: argument, contain key(name of field) and value, has the form of



The response body will contain record: fields(key/name of field and its value), id of record in table and createdTime.

This method will return the response body as *String* if created successfully else *null*.

Reference: <https://airtable.com/developers/web/api/create-records>

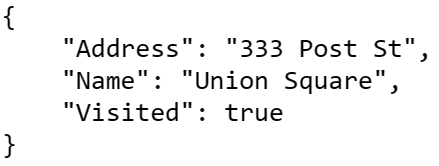
* + *updateRecord(fields : JsonObject, recordId : String, tableId : String, baseId : String, personal\_access\_token : String)*: method, used for updating single record (use HTTP request)
    - *fields(JsonObject)*: has the form that similar to argument “fields” in method *createRecord()*

Reference: <https://airtable.com/developers/web/api/update-record>

* + *equals(fields : JsonObject, fieldsList : List<Field>)*: class method, used to compare 2 records. If 2 records are the same, return *true* else *false*
* *Table*:
  + *id(String)*: your tableId
  + *name(String)*: your table name
  + *fields(List<Field>)*: list of fields (*Field* object) in a table
  + *records(List<Record>)*: list of records (*Record* object) in a table
  + *Table(table : JsonObject, baseId : String, personal\_access\_token : String)*: constructor, convert a table in AirTable to *Table* object
    - *Table(JsonObject)*: argument, has the form of

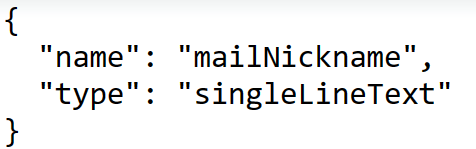


* + *syncRecord(baseId : String, personal\_access\_token : String)*: class method, sync all records in a AirTable table to *Table* object (*records(List<Record>)*)
  + *listTables(baseId : String, personal\_access\_token : String)*: method, used for listing all tables in base has id = *baseId* (use HTTP request).  
    The method will return response body as *String*
  + *pullRecord(fields : JsonObject, baseId : String, token : String)*: method, used for pulling record to table in AirTable (use HTTP request)
    - *fields(JsonObject)*: argument, contain key(name of field) and value, has the form of



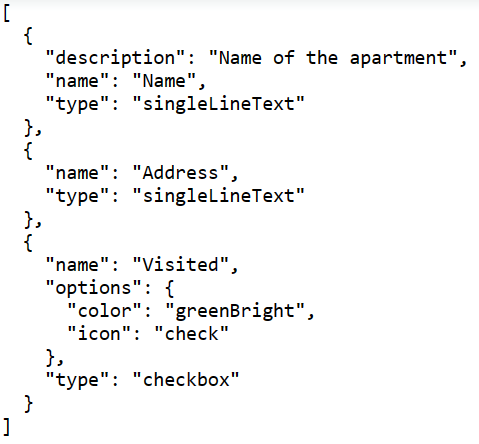
This method will return *true* if pulled successfully else *false*

* + *pullAllRecords(fields : List<JsonObject>, baseId : String, personal\_access\_token : String)*: method, used for pulling multiple records to table in AirTable.  
    This method will return *true* if pulled successfully else *false*
  + *addField(field : JsonObject, baseId : String, token : String)*: method, used for add new field to table.
    - *field(JsonObject)*: argument, has the form of



This method will return *true* if added successfully else *false*.

* + *updateRecord(fields : JsonObject, record : Record, baseId : String, personal\_access\_token : String)*: method, used for updating record in AirTable table (similar to method *Record.updateRecord*) and record in Table object (*records(List<Record>)*). Return *true* if successful else *false*
  + *getTable(tableId : String, baseId : String, personal\_access\_token : String)*: method, used for getting table information (*name*, *fields*,…) that has id = *tableId* from a base (*baseId*). Return as *JsonObject*
  + *writeTableToXLSX(filename : String, baseId : String, personal\_access\_token : String)*: method, used for writing the data in a table (AirTable) to XLSX file.
  + *addRecord(fields : JsonObject, record : Record, baseId : String, personal\_access\_token : String)*: method, used for adding record in AirTable table (similar to method *Record.addRecord*) and record in Table object (*records(List<Record>)*). Return true if successful else false
  + *dropRecord(fields : List<JsonObject>, baseId : String, personal\_access\_token : String)*: method, used for dropping all records that exist in a Table object (*records(List<Record>)*) but *fields* (argument) *(List<JsonObject>)*.
  + *createTable(name : String, fields : JsonArray, baseId : String, personal\_access\_token : String)*: method, used for creating a new table and returning the schema for the newly created table.
    - *fields(JsonArray)*: argument, has the form of

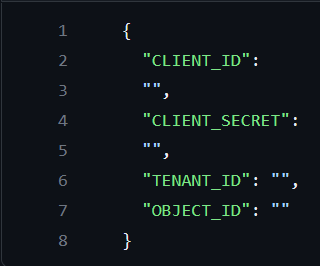


This method will return response body as *String*.

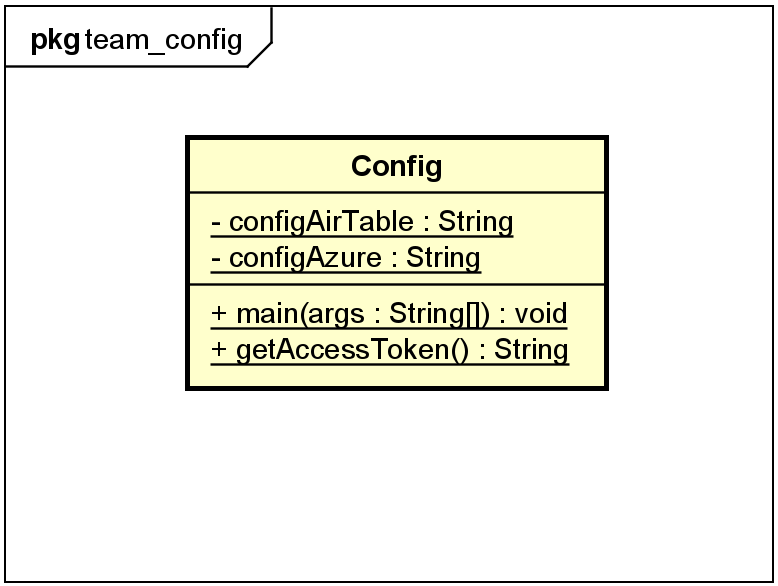
Reference: <https://airtable.com/developers/web/api/create-table>

### **5.2.6. *template.accessInfo***

This package doesn’t have any classes, only contains 2 file “*configAirTable.json*” and “*configAzure.json*”.

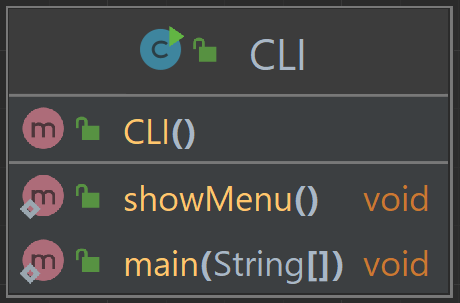
* *configAirTable.json*: Contain your *baseId* and *personal\_access\_token* of AirTable.  
  
* *configAzure.json*: Contain your Microsoft 365 *CLIENT\_ID*, *CLIENT\_SECRET*, *TENANT\_ID* and *OBJECT\_ID.*  
  

### **5.2.7. *template.team\_config***



* *Config*:
  + *configAirTable(String)*: attribute, contain absolute path of *“configAirTable.json”*
  + *configAzure(String)*: attribute, contain absolute path of *“configAzure.json”*

### **5.2.8. *CLI.java***

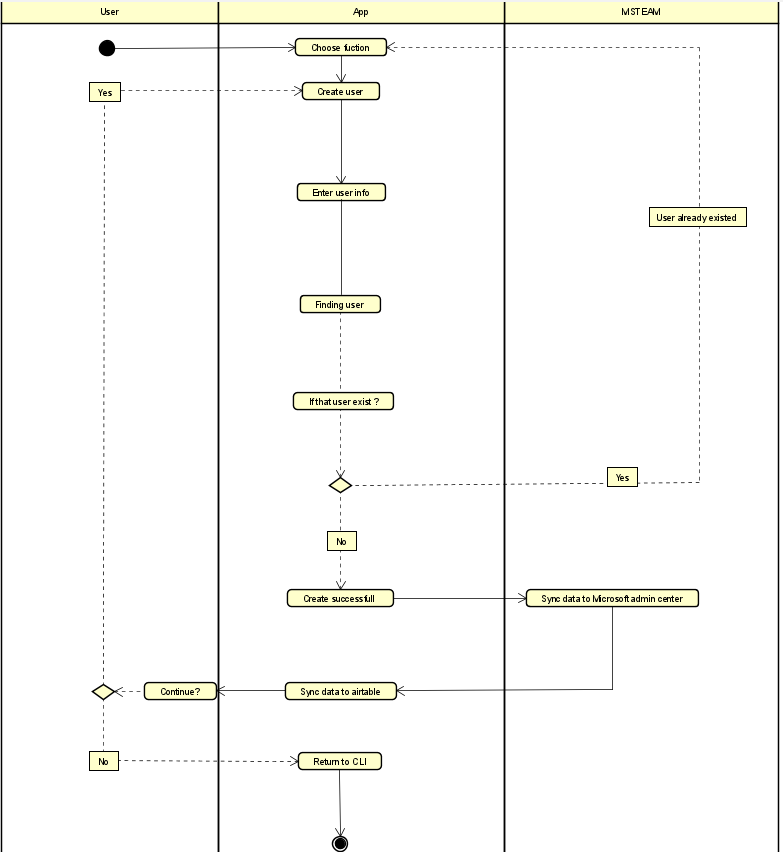


*showMenu()*: method, used to show the main menu of the program (application) on the console.

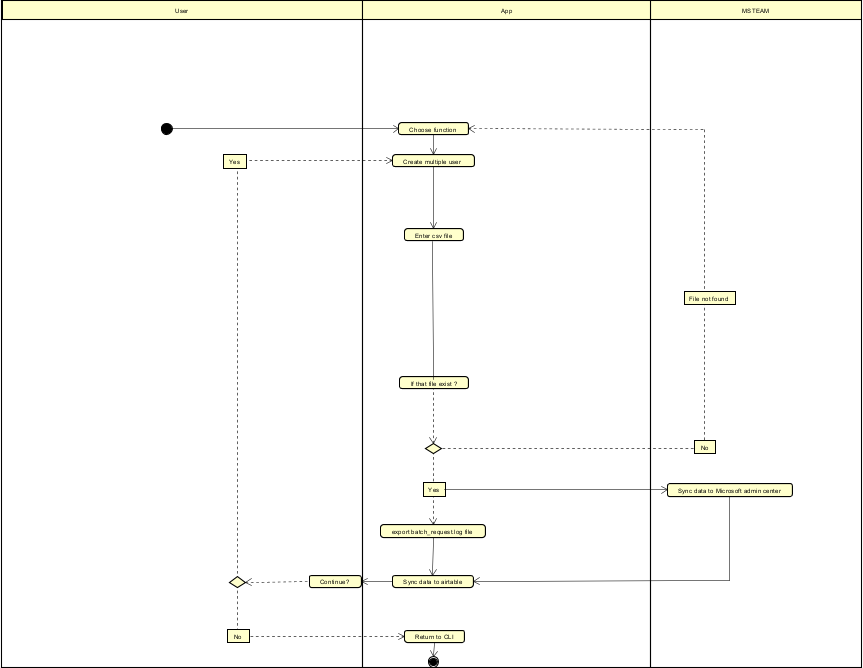
## **5.3 Activity diagram**

*a.UserService*

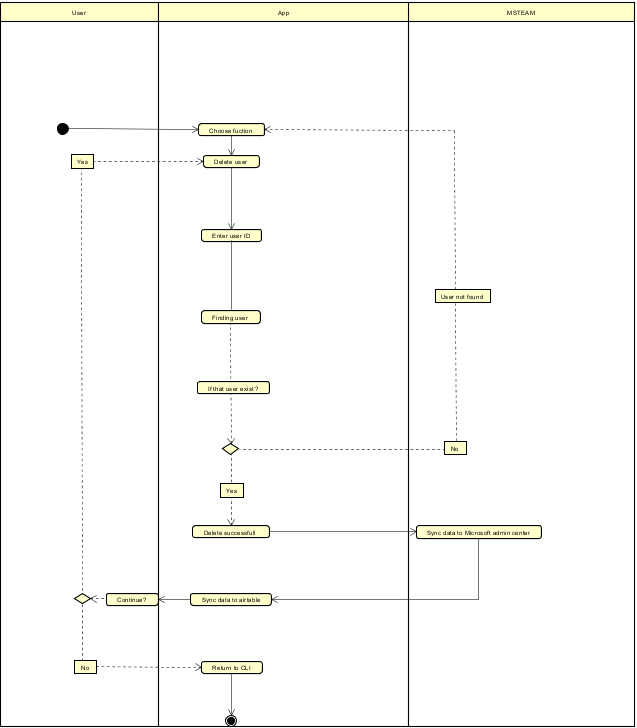
* *CreateUser:*



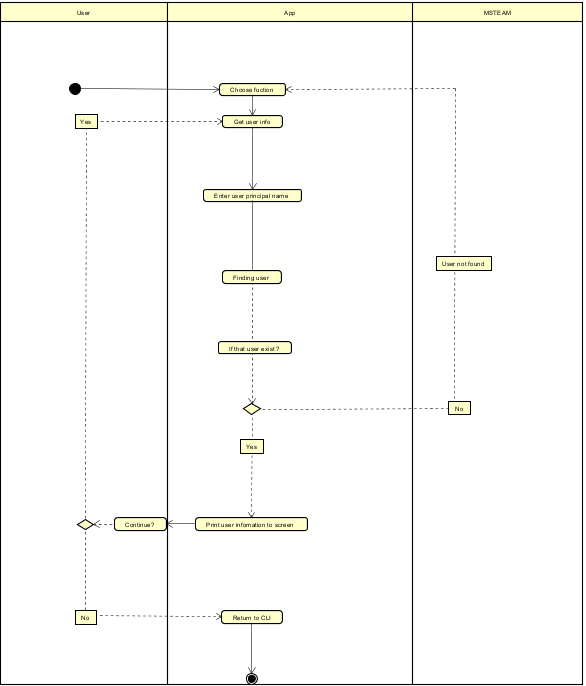
* *Create multiple users from csv file*



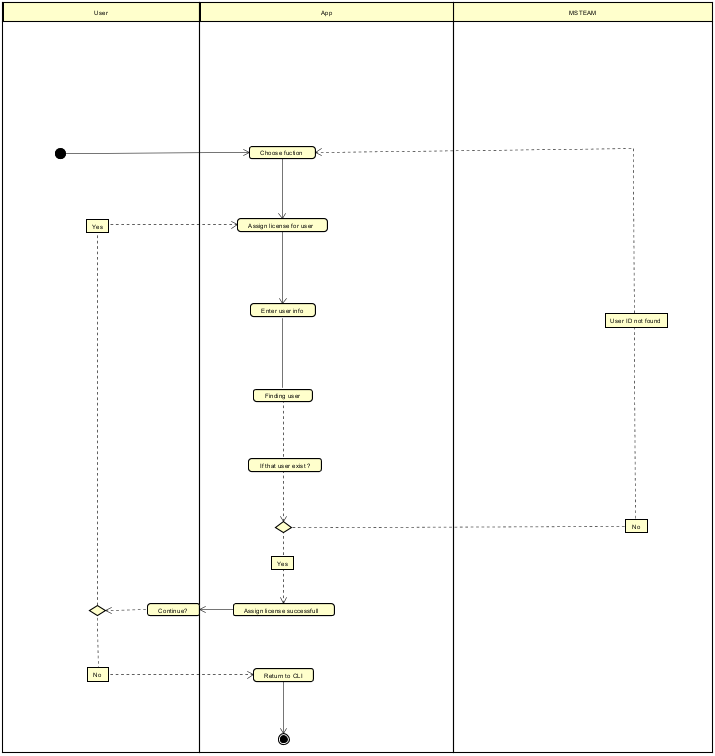
* *Delete user from organization*



* *Get user by principal name*

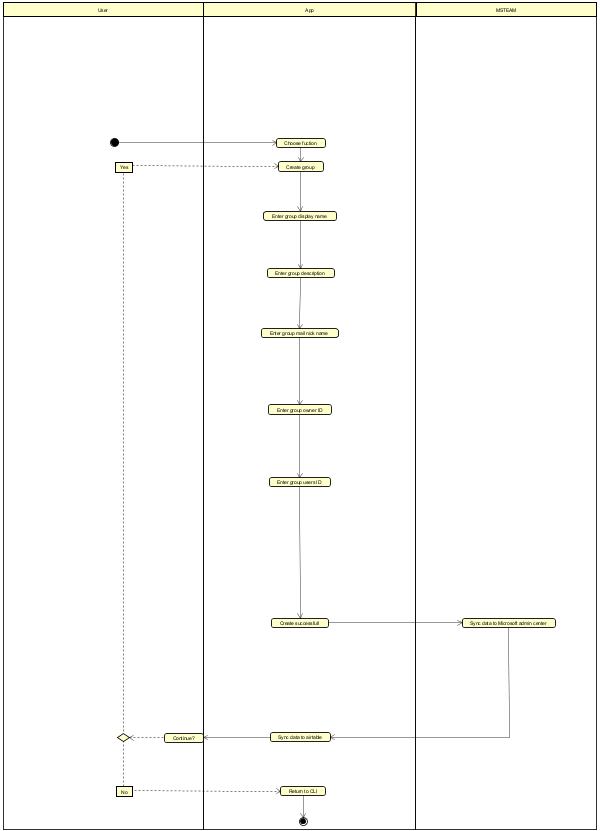


* *Assign license for user*

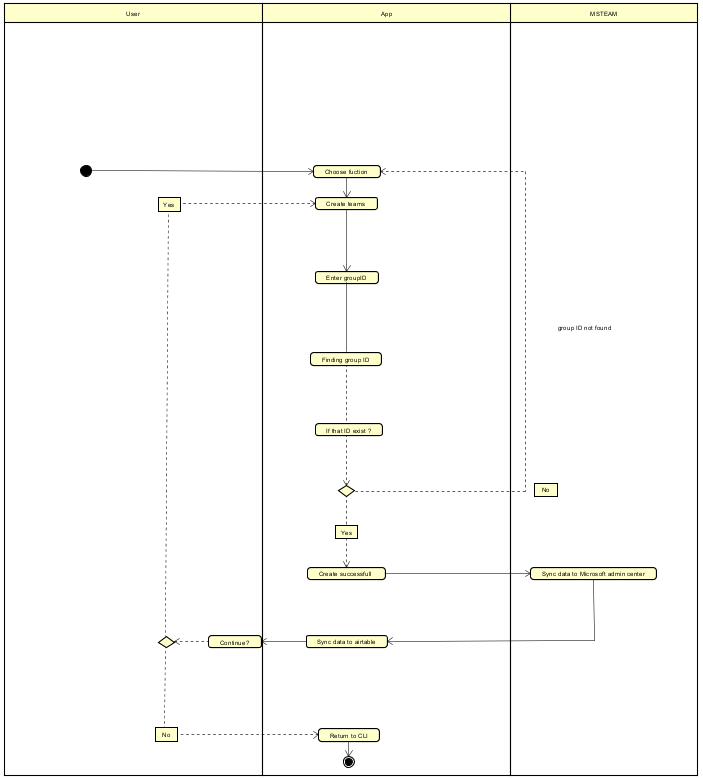


*b. GroupService*

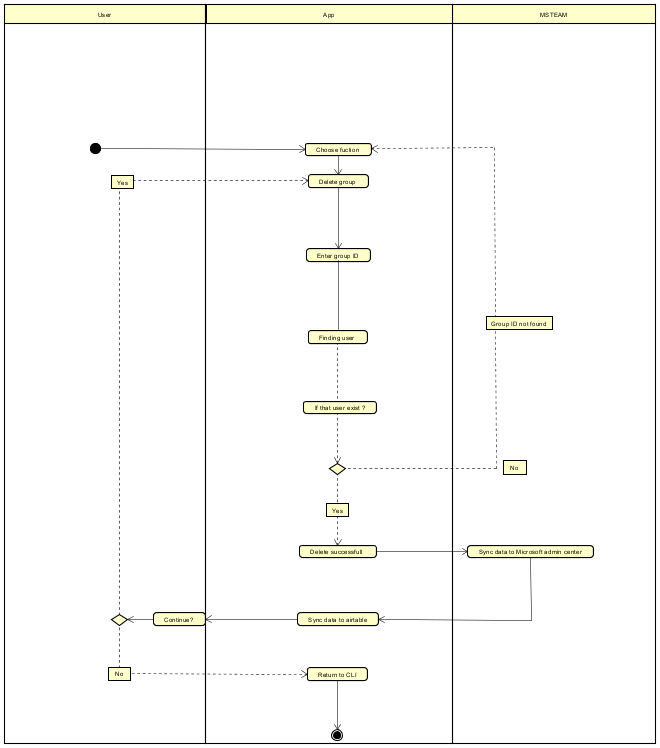
* *Create Group*



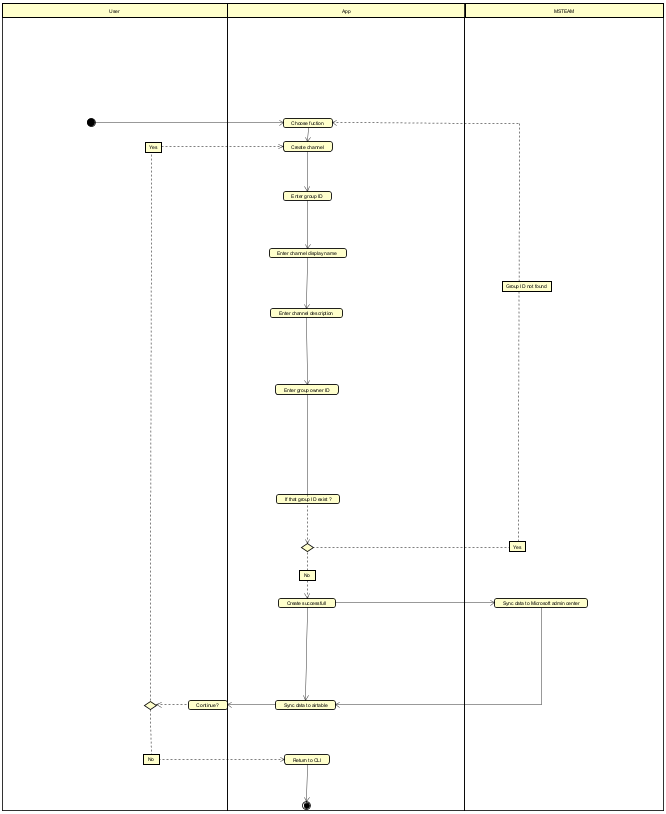
* *Create Team*



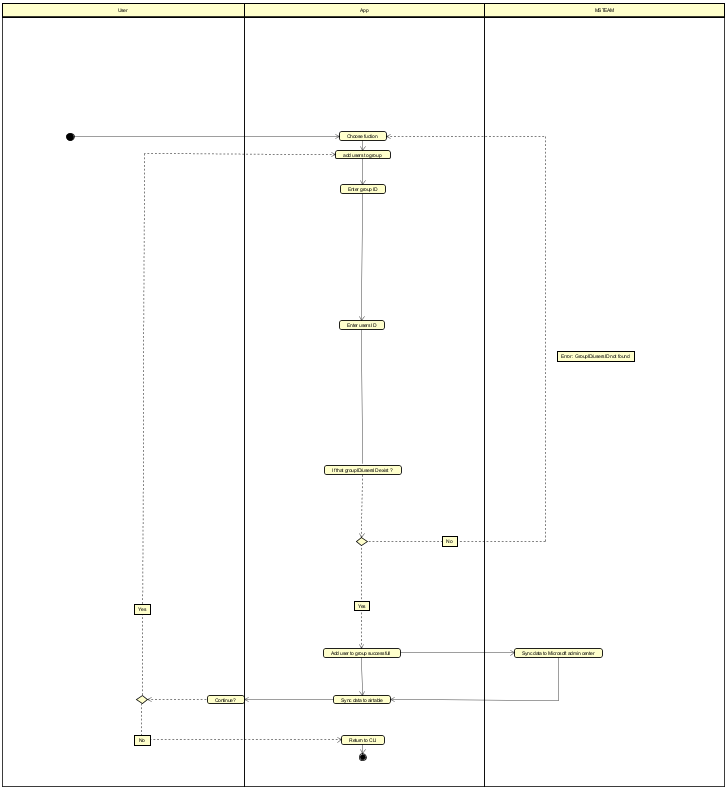
* *Delete Team(Group)*



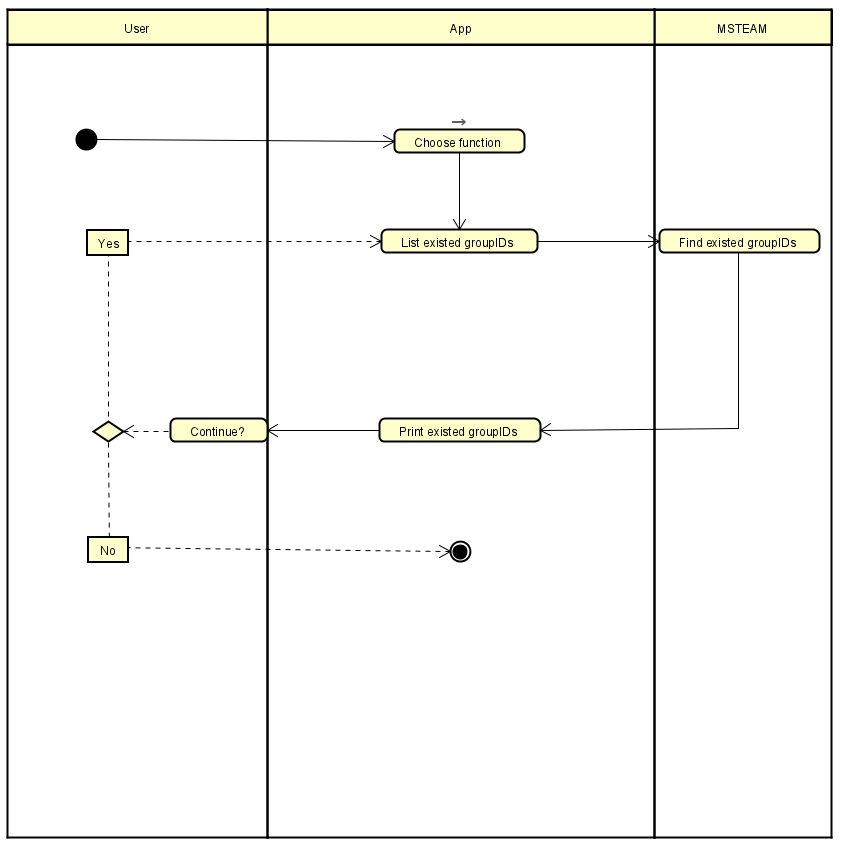
* *Create Channel*



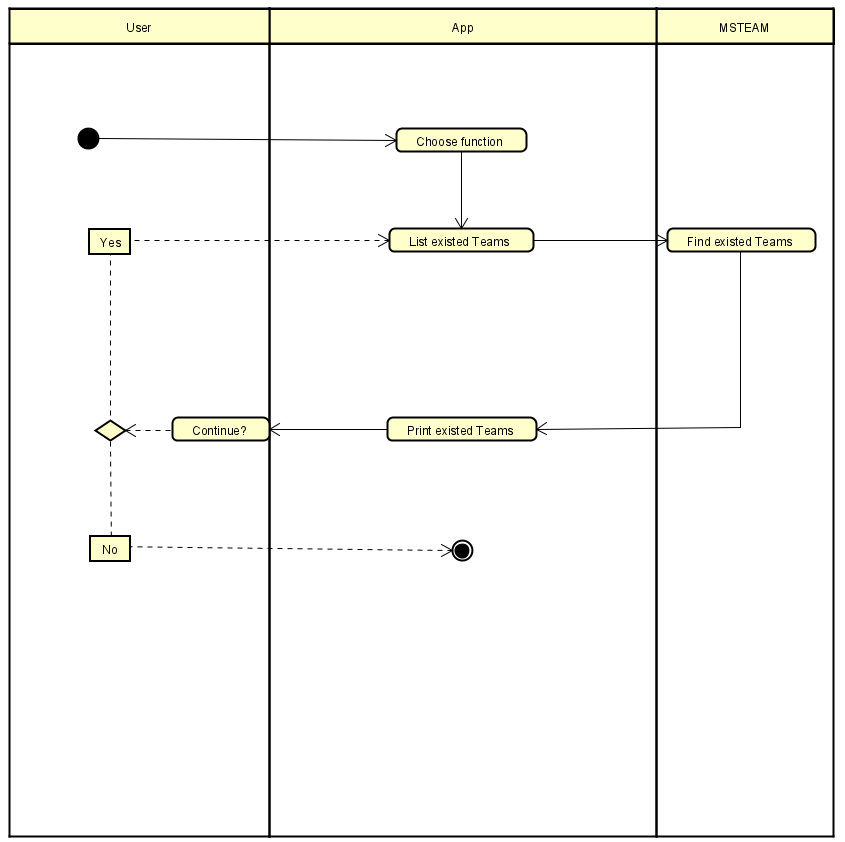
* *Add user/multiple users to Team*



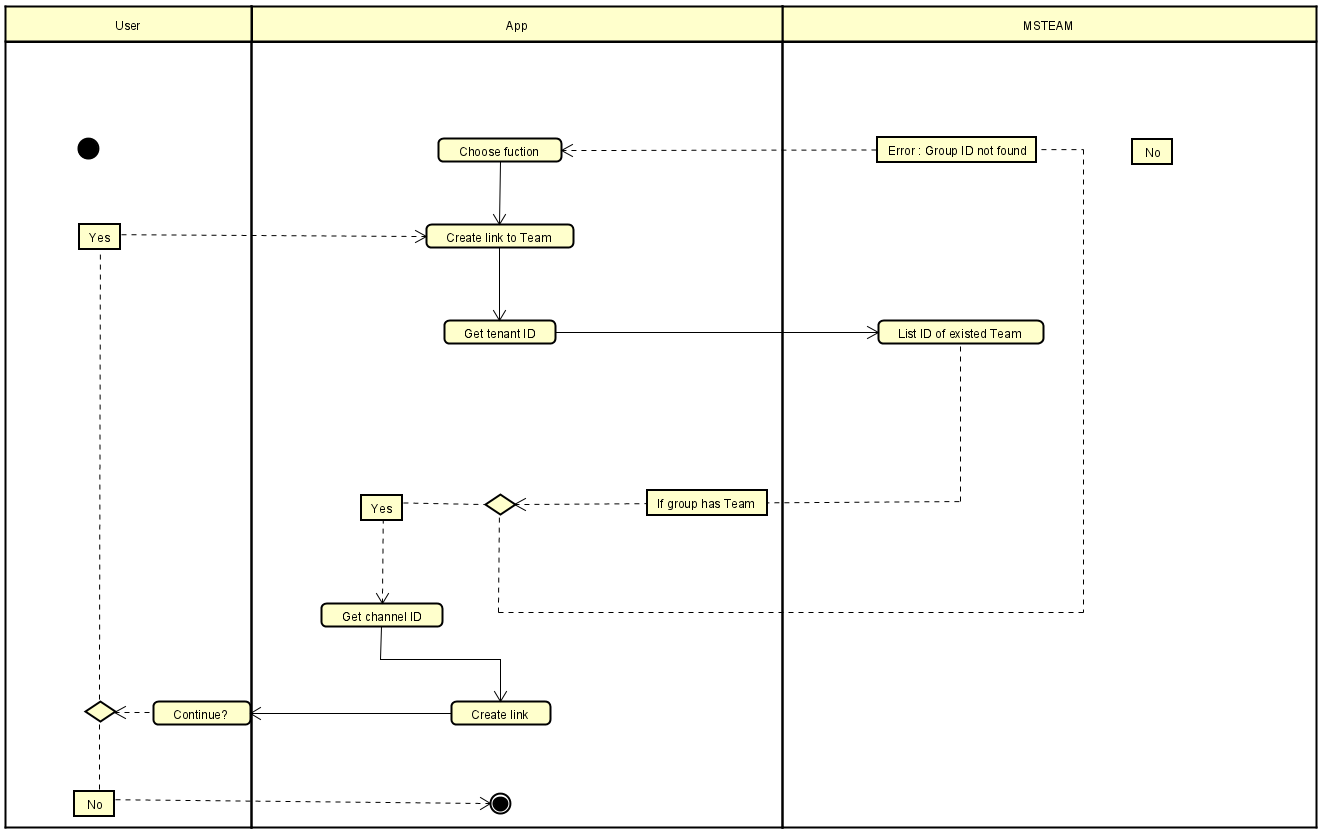
* *List existed groupIDs*



* *List existed teams*

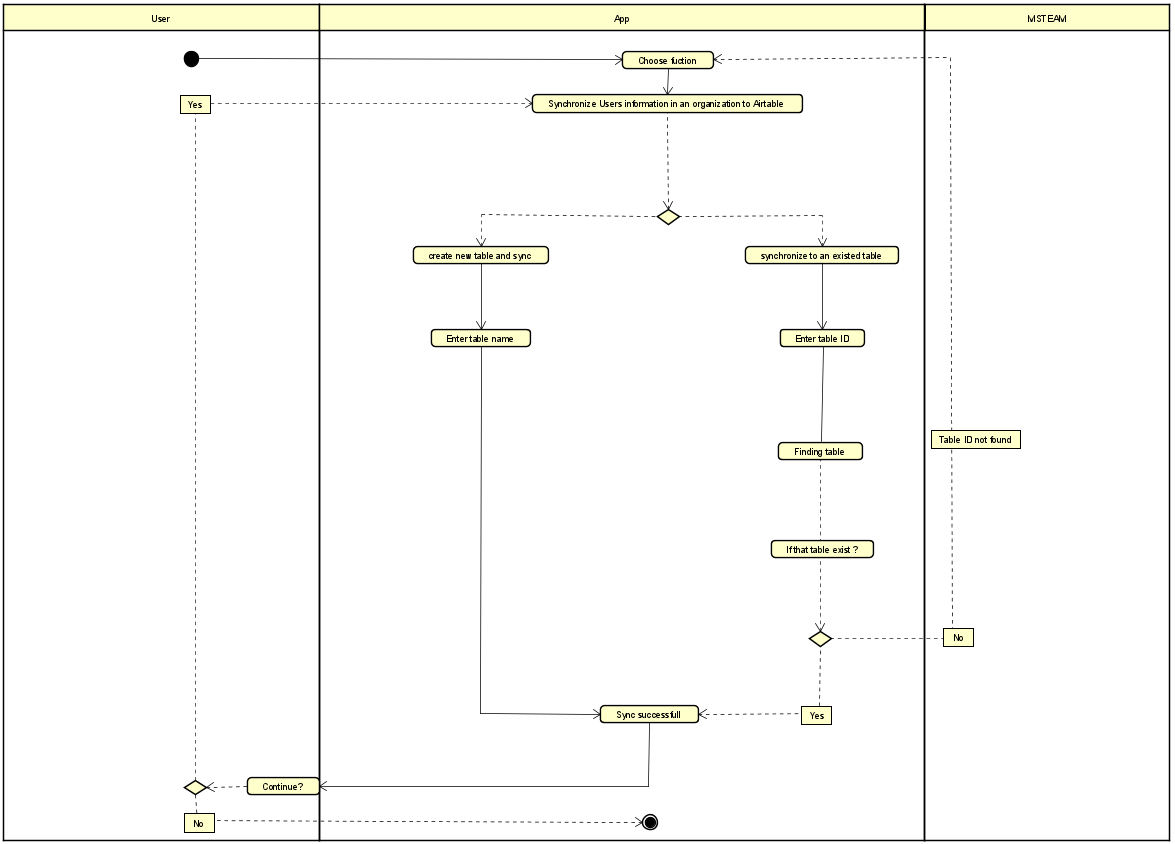


* *Create link to team*

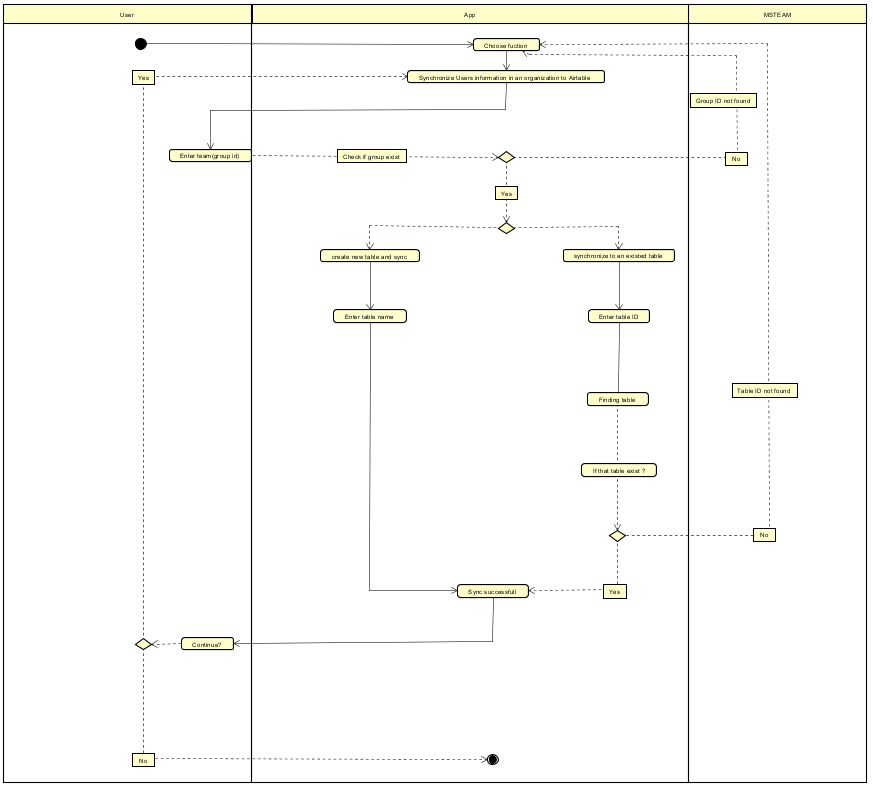


*c. AirTable:*

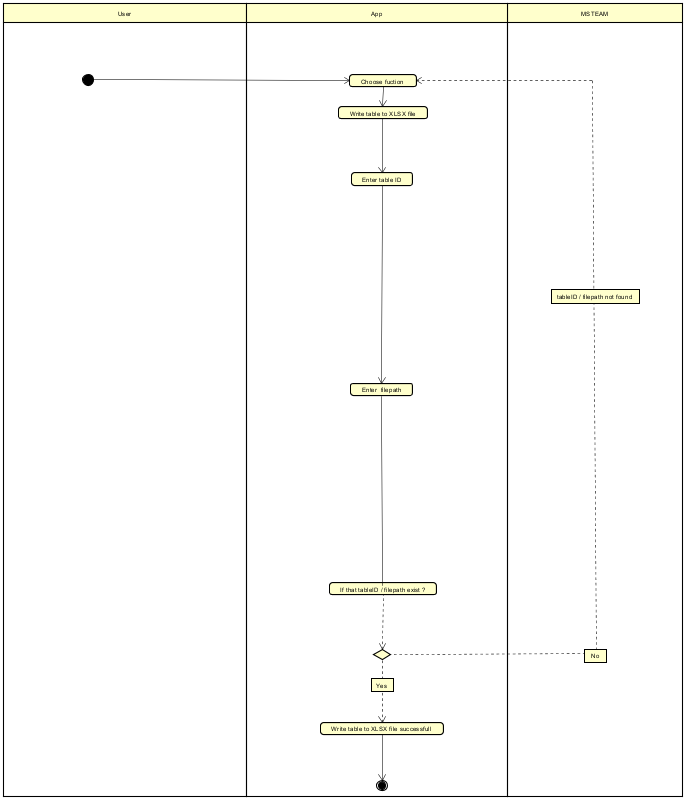
* *Synchronize Users information in an organization to Airtable*



* *Synchronize Users information in a Team to Airtable*



* *Write to XLSX file*

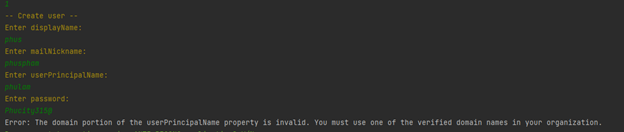


# **6. Demo and Evaluation**

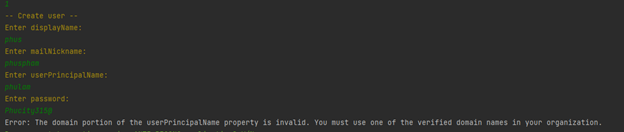
## **6.1. Demo**

1.Test create user

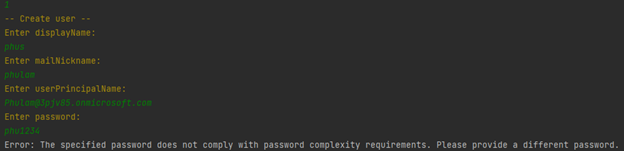
* *userPrincipalName* error
  + Wrong domain



* + Another object with the same value existed



* Password error
  + Password does not comply with password complexity requirements

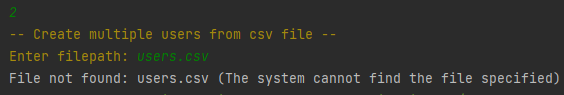


* Create success

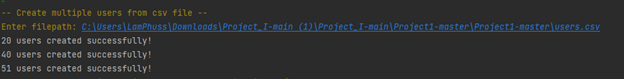


2. Create multiple users from csv file

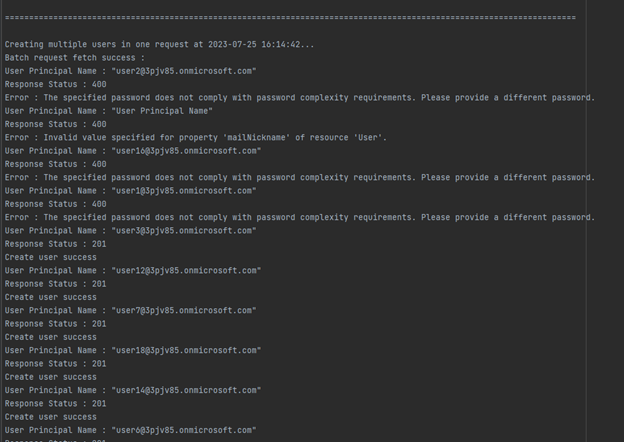
* Wrong file path



* Create success

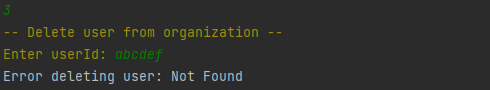


* + Diary file:

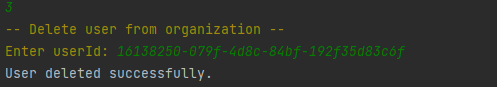


3.Test delete user

* User not found error:

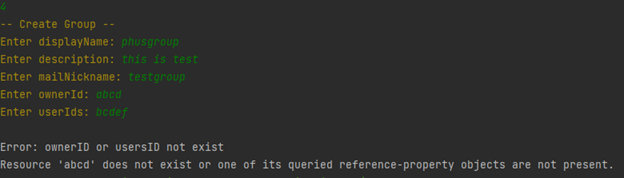


* Delete success:



4.Test create group

* *userID* or *ownerID* not exist:



* Create group success:

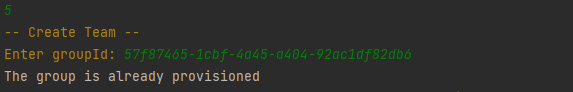


5.Test create team

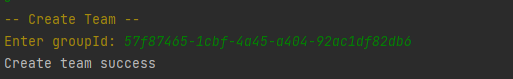
* Wrong group ID format error:



* Group already created:



* Create team success:

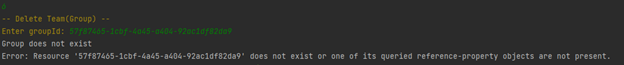


6.Delete team

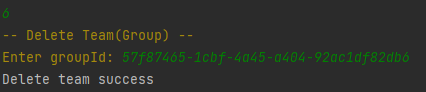
* Wrong group ID format:



* Group not found:



* Delete team(group) success:

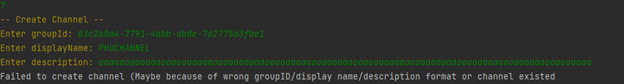


7. Create channel

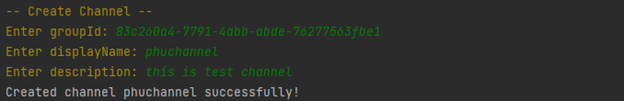
* Group not found:



* Channel existed or wrong *groupId*/*displayName*/*description* format:

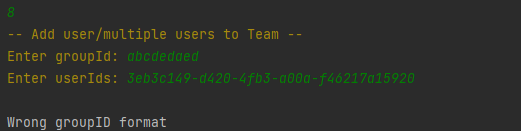


* Create channel success:

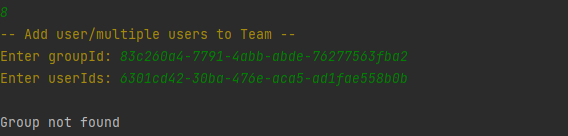


8.Add user/multiple users to Team

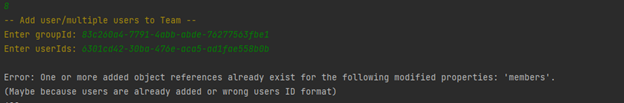
* Wrong *groupId* format:



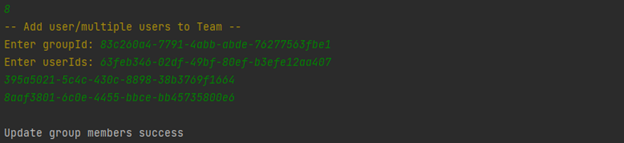
* Group not found:



* User are already added or wrong *users ID* format:

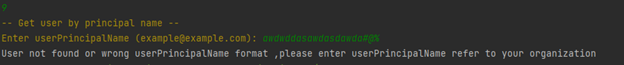


* Add user success:

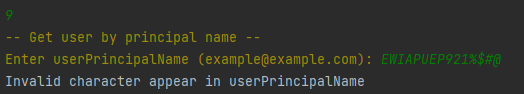


9. Get user by principal name

* User not found or wrong *userPrincipalName* format:



* Invalid character appear in user principalname (example : $):

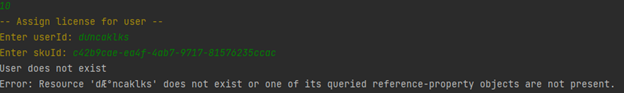


* Get user success:



10. Assign license for user:

* User not found:



* SkuID (license id) not found:



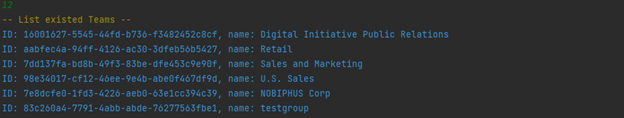
* Assign success:



11. List existed groupIDs:



12. List existed Teams



13. Create link to team

* Invalid group ID :



* Group not found :



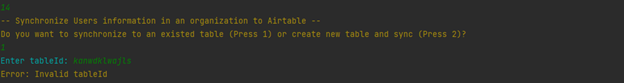
* Create success:



14.

a. If you press 1:

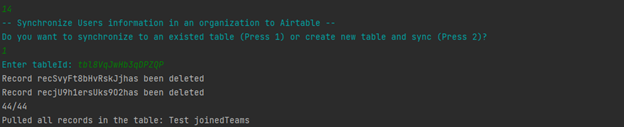
* Invalid table ID:



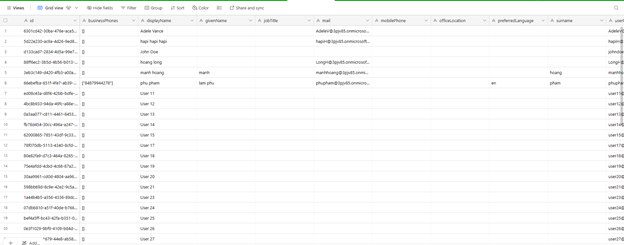
* Invalid table field (table field may not contain user attributes):



* Pull all records success:

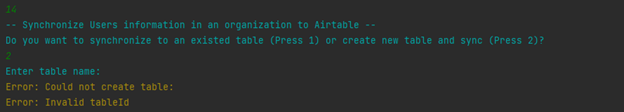


Result:



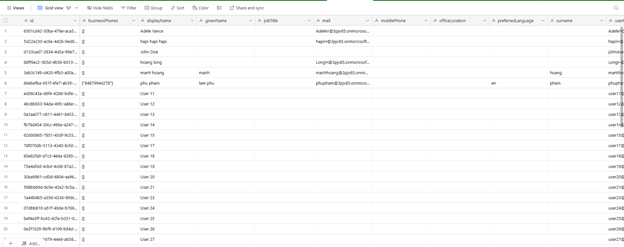
b. If you press 2:

* Table name is null:



* Create table and pull all record success:

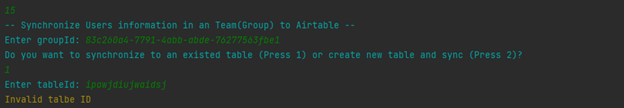


Result:  


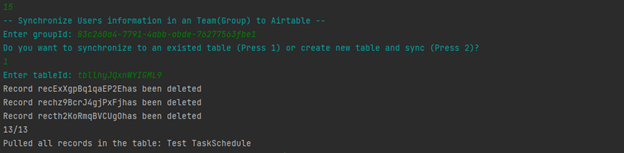
15.

a. If you press 1:

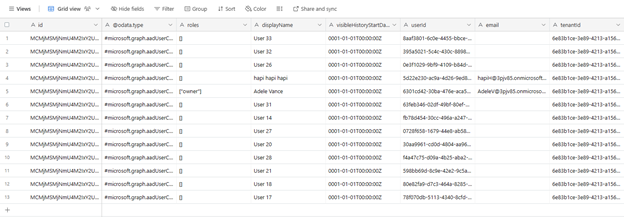
* Invalid table ID:



* Sync all records from the given group with success:

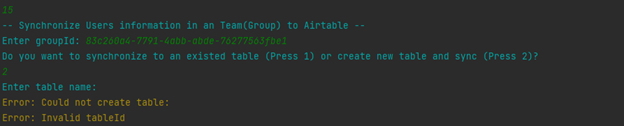


Result:

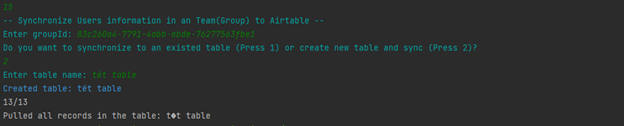


b. If you press 2:

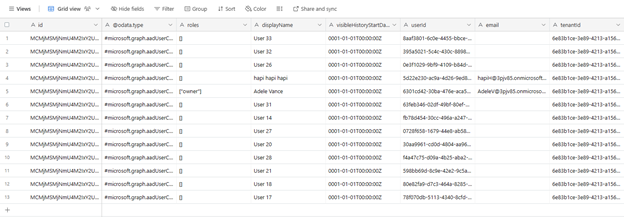
* Table name is null



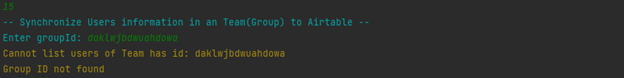
* Create table and sync all records from the given group with success:



Result:

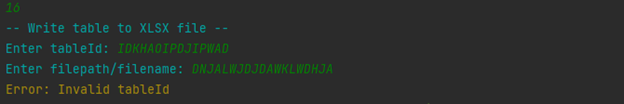


c. *groupId* not found:

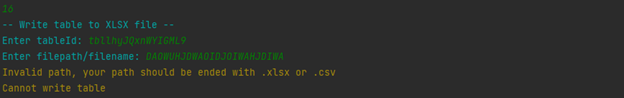


16. Write table to XLSX file:

* Invalid table ID:



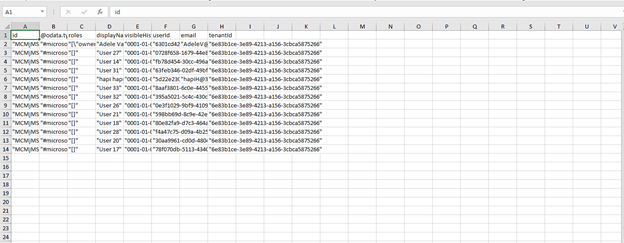
* Invalid path:



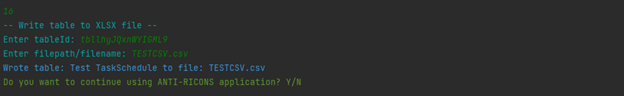
* Write table to xlsx file success:



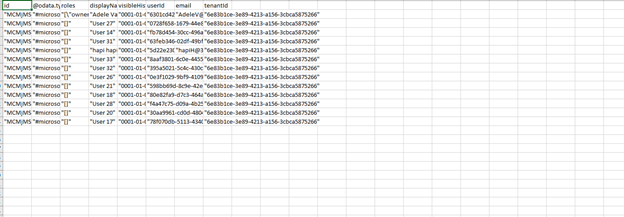
Result:



* Write table to csv file success:



Result:

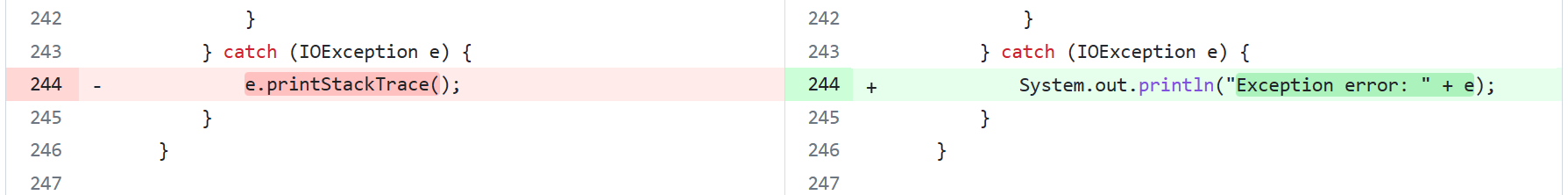


## **6.2. Evaluation**

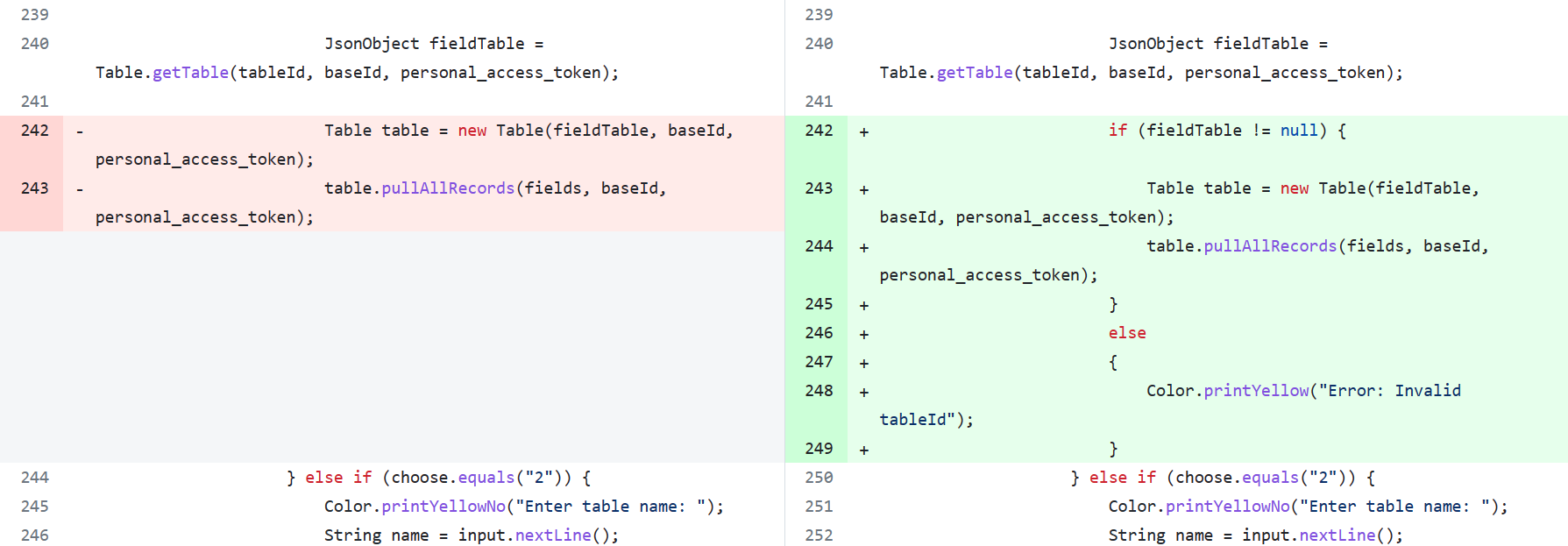
### **6.2.1. Check by SonarQube (static code)**

a. Bugs

b. SecurityHotspot

* Not handle exception  
    
    
  Using *printStackTrace()* to print the stack trace of the exception (detailed information about the exception can potentially reveal sensitive information. Using *println("Exception error: " + e)* (print error message) instead of *e.printStackTrace()* to notify user.
* Configuring loggers (**fixed**)  
    
    
    
  Reference: <https://rules.sonarsource.com/java/RSPEC-4792/>

### **6.2.2. Dynamic code analysis**

* Not handle exception  
    
  
* If *fieldTable(JsonObject)* is not checked before using, it can lead to *NullPointerException* because method *Table.getTable()* can return *null* value.

### **6.2.3. Secure coding**

* Input Validation (Limiting User Input): The code checks the user's input for various types of selections, such as menu options, file paths to prevent any unwanted behavior or execution of code. It limits the user's choices to a set of predefined integer values and valid strings, such as *tableId (AirTable)* and *groupId (MS 365)*.
* Exception Handling: The code includes exception handling to try and catch potential exceptions that may occur when run the program, control the error message (should not use *e.printStackTrace()*). A stack trace (detailed information) should never be visibile to end users (for user experience and security purposes)
* Configuring Logger: Configuring *java.util.Logger* is security-sensitive. It can led to some vulnerabilities. Reference: <https://rules.sonarsource.com/java/RSPEC-4792/>

# **7. Configuration guide**

* **Information**
  + Add your access information to these json files:
    - Path: "*src/main/java/infoTaskSchedule.json*" contains *groupId* (MS Team), *tableId* (AirTable) for automatical synchronous feature.
    - Path: "*src/main/java/template/accessInfo/configAzure.json*" contains *CLIENT\_ID, CLIENT\_SECRET, TENANT\_ID, OBJECT\_ID* (Microsoft Graph)
    - Path: "s*rc/main/java/template/accessInfo/configAirTable.json*" contains *baseId*, *personal\_access\_token* of AirTable
* **Configuration**
  + Download maven from <https://maven.apache.org/download.cgi> , then config this file in edit system environment variables (if you use window) to run the task schedule
  + Change all the path of json file in the code to your current absolute path (In *Config.java*, you need to change the value of variable "*configAirTable*" to the absolute path of file named "*configAirTable.json*", the value of variable "*configAzure*" to the absolute path of file named "*configAzure.json*")
  + Note: you dont need to config file *configAzure.json* since this file already config
* **Set up automatical synchronous feature**
  + Step 1: open "laplich.bat" file in the repository and change the path to your java.exe, and change the path to "TaskScheduleForAntiRicons" class file  
    ##### You can config task schedule for 2 ways  
    ###### First way
  + Step 2: open the task scheduler in the window search box
  + Step 3: on the right hand box, click Create Task
  + Step 4: enter the name of the task
  + Step 5: at the Security options, choose "Run whether user is logged on or not" and select the "Do not store password ....". Finally, select "Run with highest privileges" box
  + Step 6: at the Triggers window, click New and a New Trigger window will be popped up
  + Step 7: at Settings section, select "Daily"
  + Step 8: on the right handside of the Settings section, Change the time to what you desired and click "OK"
  + Step 9: at the Actions wintdow, click New and a New Action window will be popped up
  + Step 10: at the Action box, choose Start a program
  + Step 11: at the Program/script box, enter the path to the "Syncronize.bat" file
  + Step 12: hit "OK"
  + Step 13: at the Actions section on the right handside click "Enable All Tasks History". Now every changes of this task will be logged at the History section

###### Second way

* + Step 2: open cmnd , write this script schtasks /create /tn "MyNewTask" /tr "C:\Users\LamPhuss\Desktop\laplich.bat" /sc daily /st 08:00 /ru "SYSTEM" (change the path and /tn : for task name and /sc for frequency of the task and /st hh:ss for time to execute task

**Github URL for source code**: <https://github.com/sangnguyenthien/Project_I>

**Contact**: Nguyen Thien Sang 20214972 – Email: [sang.nt214972@sis.hust.edu.vn](mailto:sang.nt214972@sis.hust.edu.vn)

Pham Lam Phu 20214969 – Email : [phu.pl214969@sis.hust.edu.vn](mailto:phu.pl214969@sis.hust.edu.vn)