

Pridar Chatbot

Developer Manual

Adel Mouffok

Overview

Overview

Main Functions:

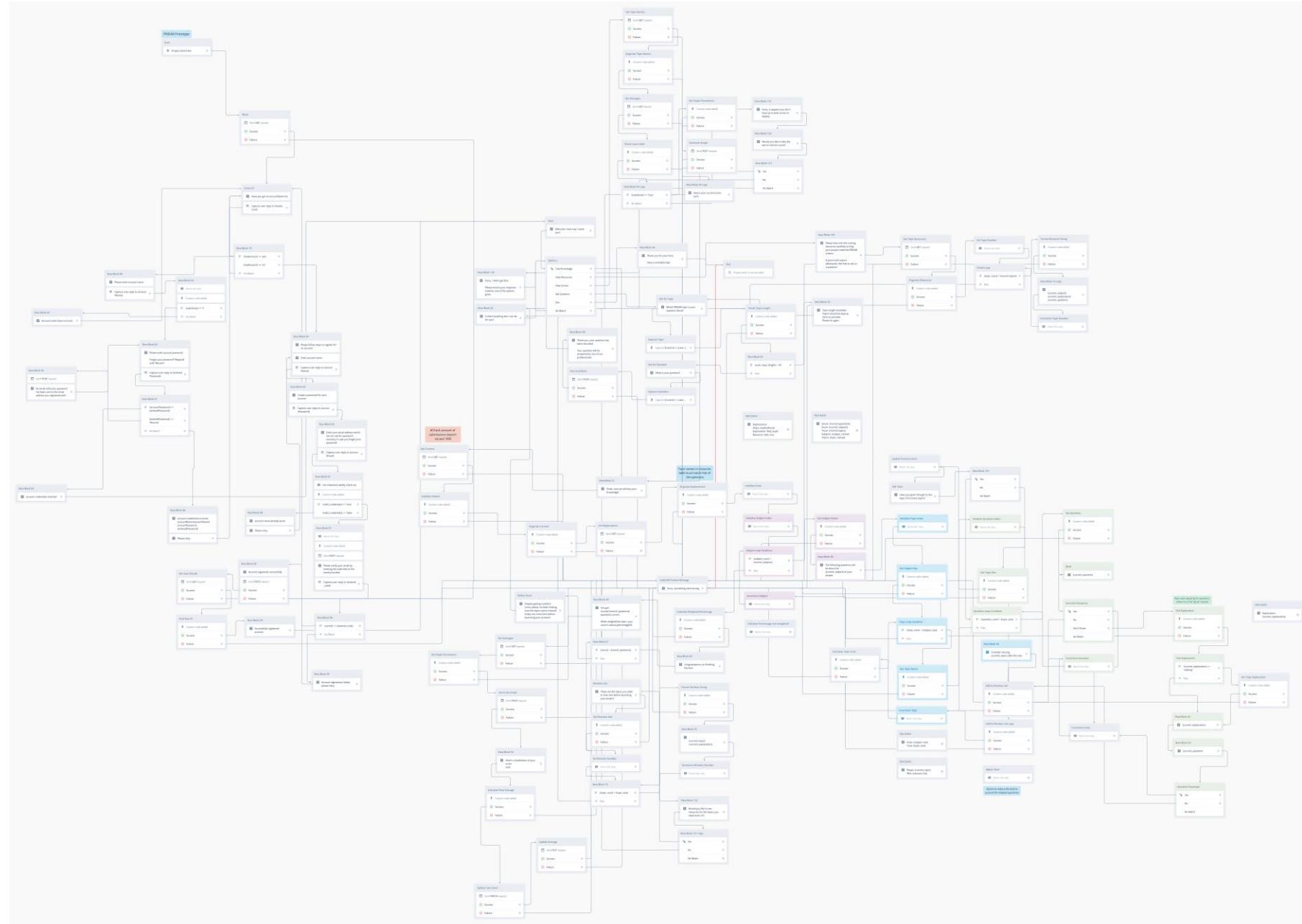
- Login/register
- Test
- Q/A
- View Scores
- View Resources

Databases:

- Jotform
- Airtable

Integrations:

- Sendgrid
- QuickChart



Overview

Main Functions:

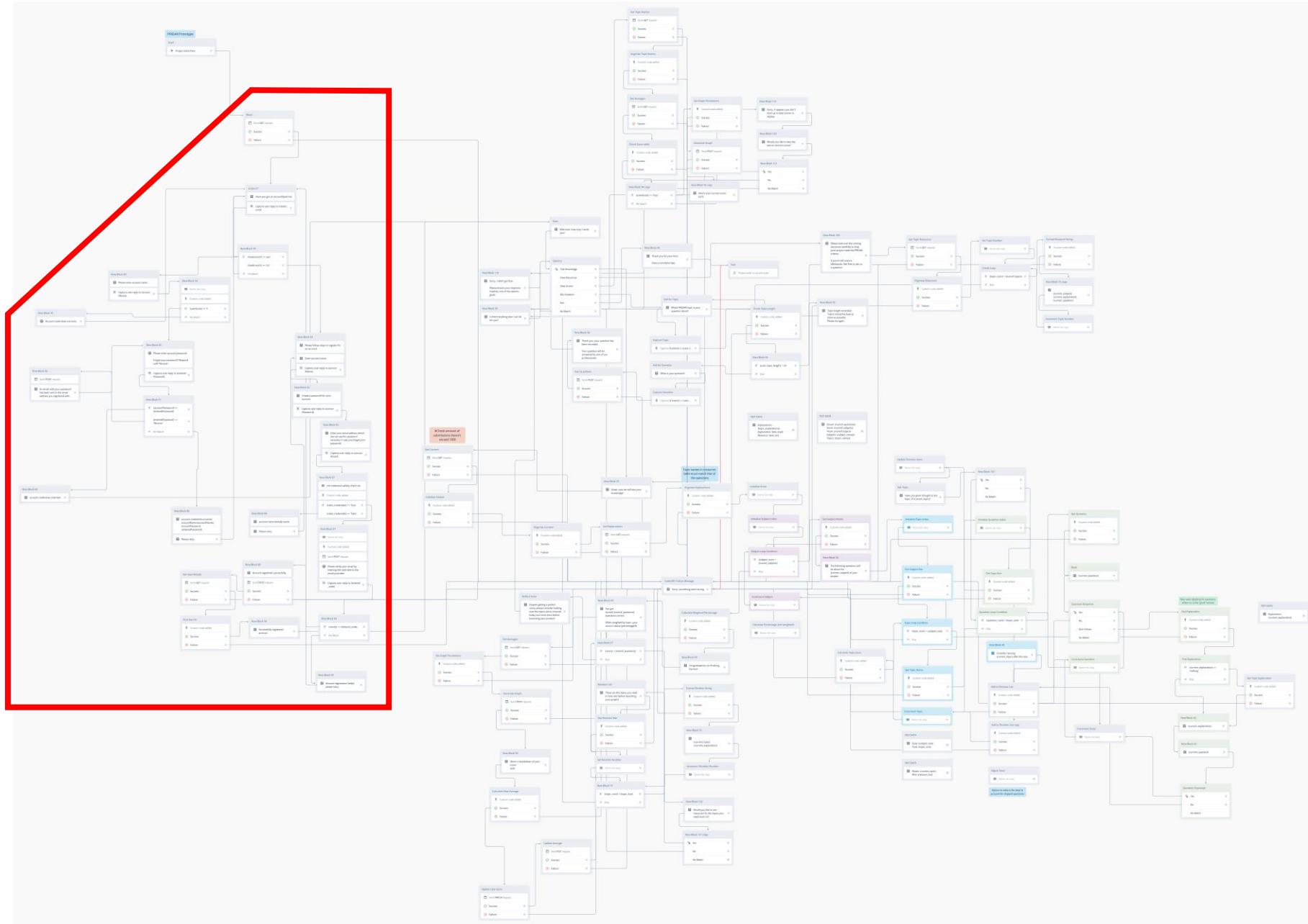
- **Login/register**
- Test
- Q/A
- View Scores
- View Resources

Databases:

- Jotform
- Airtable

Integrations:

- Sendgrid
- QuickChart



Overview

Main Functions:

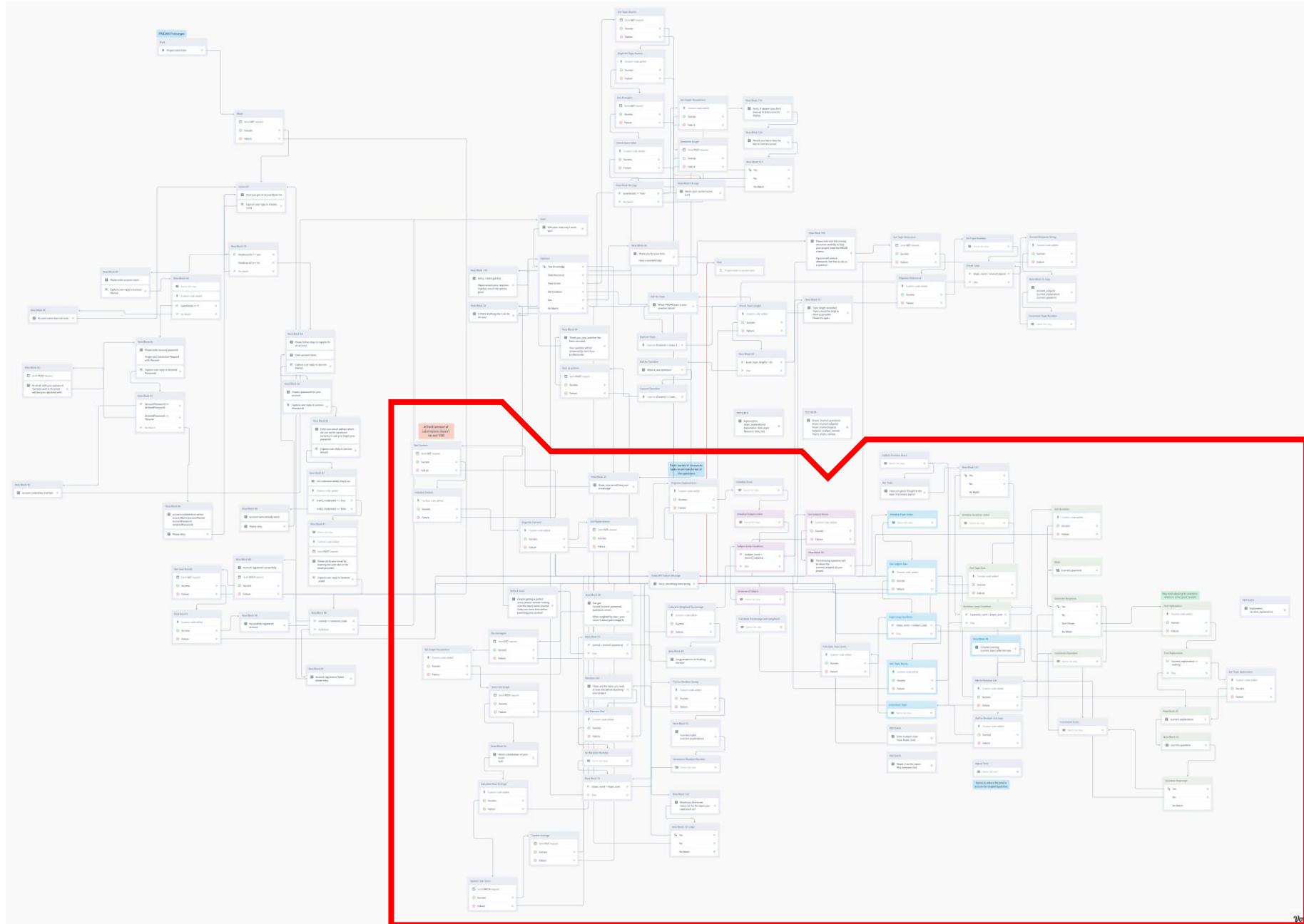
- Login/register
- **Test**
- Q/A
- View Scores
- View Resources

Databases:

- Jotform
- Airtable

Integrations:

- Sendgrid
- QuickChart



Overview

Main Functions:

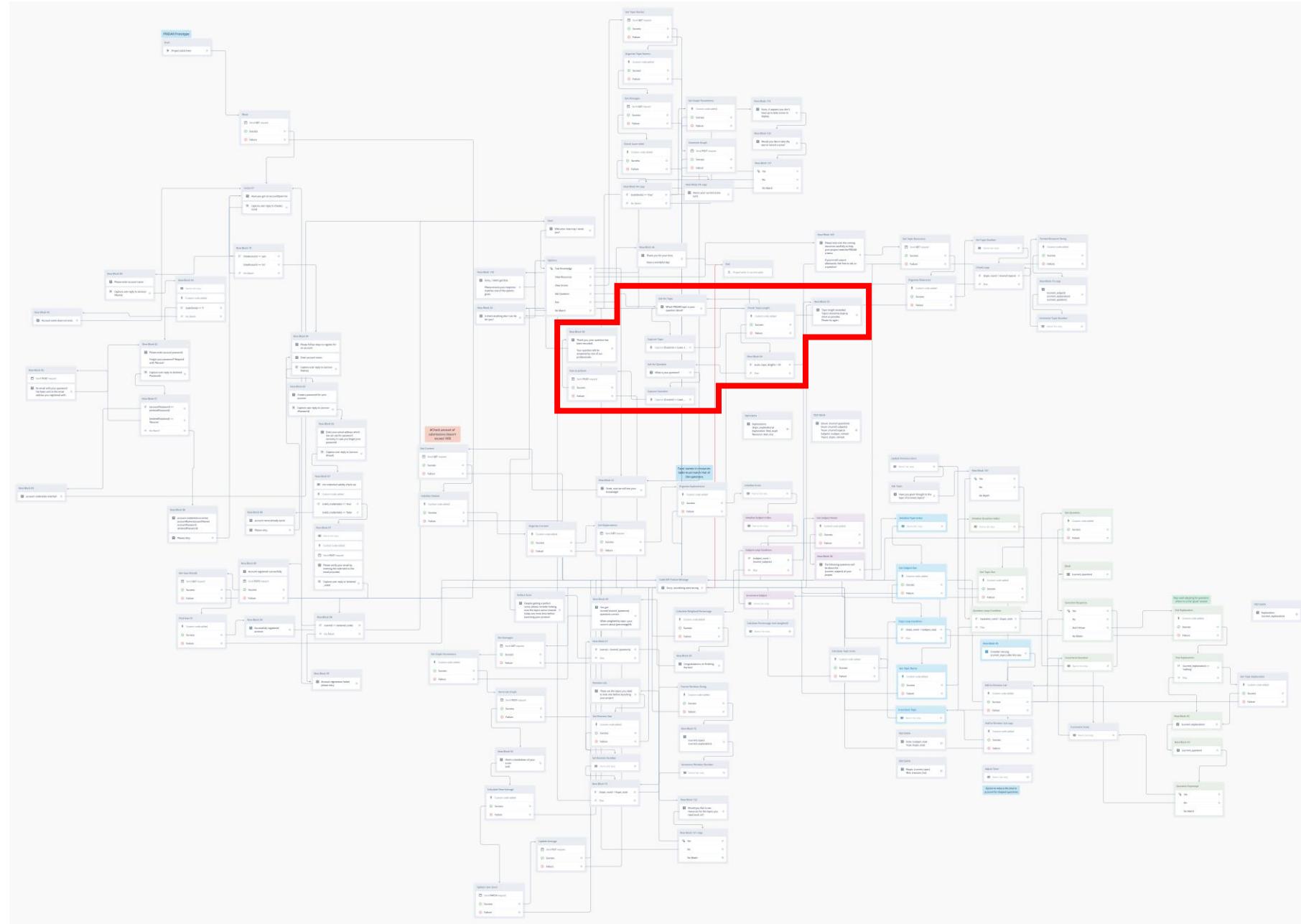
- Login/register
- Test
- **Q/A**
- View Scores
- View Resources

Databases:

- Jotform
- Airtable

Integrations:

- Sendgrid
- QuickChart



Overview

Main Functions:

- Login/register
- Test
- Q/A
- **View Scores**
- View Resources

Databases:

- Jotform
- Airtable

Integrations:

- Sendgrid
- QuickChart

Overview

Main Functions:

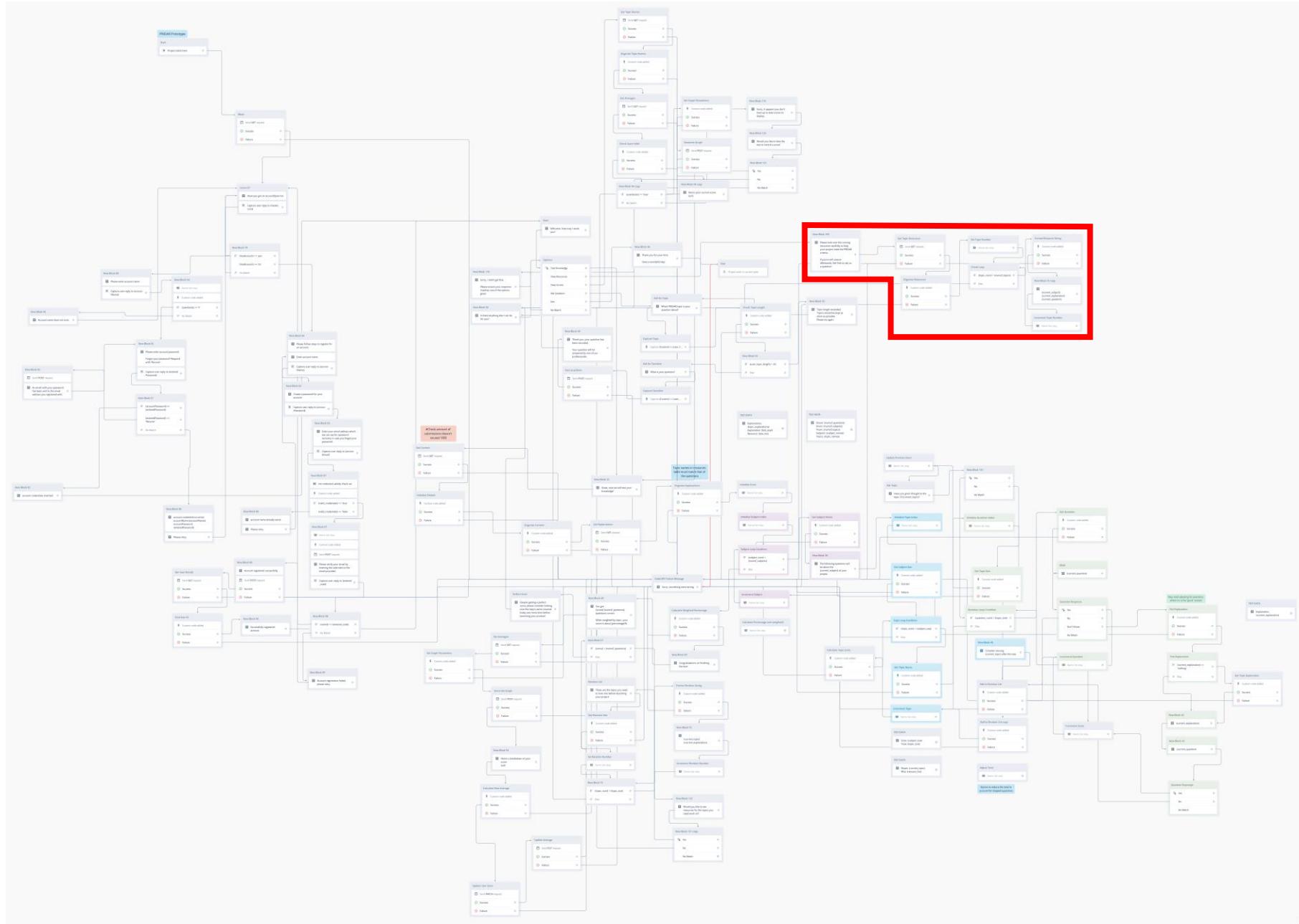
- Login/register
- Test
- Q/A
- View Scores
- **View Resources**

Databases:

- Jotform
- Airtable

Integrations:

- Sendgrid
- QuickChart



Overview

Main Functions:

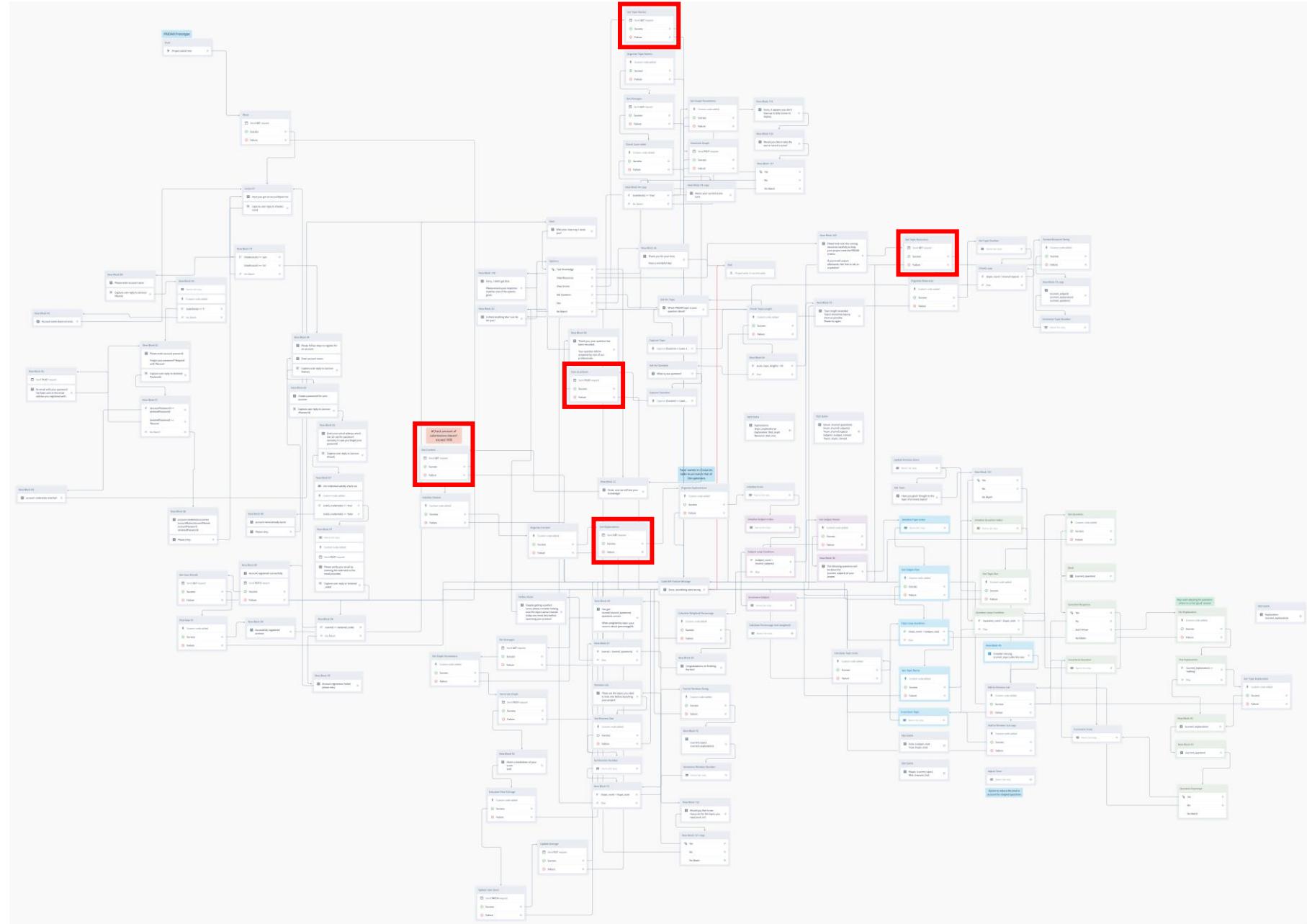
- Login/register
- Test
- Q/A
- View Scores
- View Resources

Databases:

- Jotform
- Airtable

Integrations:

- Sendgrid
- QuickChart



Overview

Main Functions:

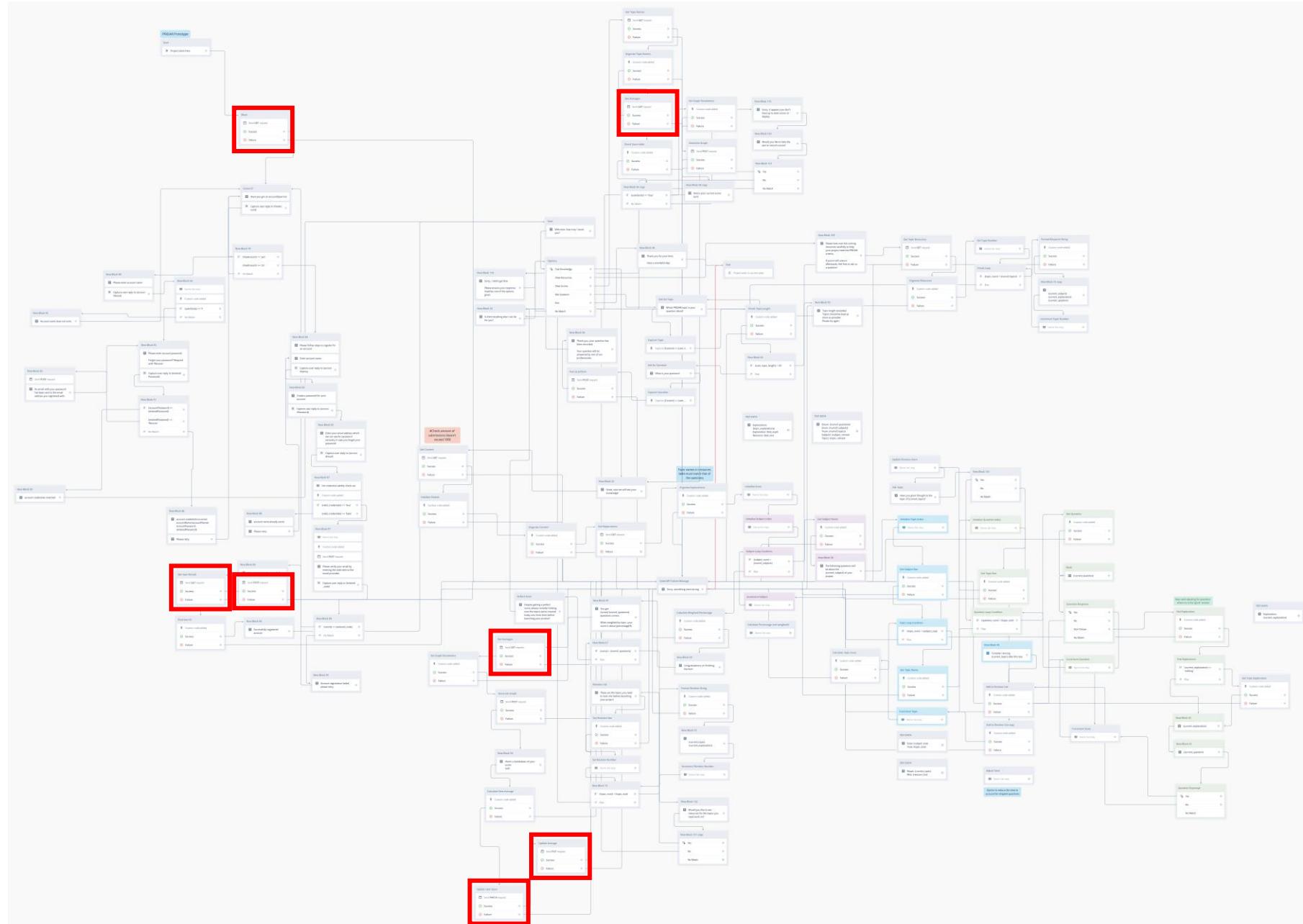
- Login/register
- Test
- Q/A
- View Scores
- View Resources

Databases:

- Jotform
- **Airtable**

Integrations:

- Sendgrid
- QuickChart



Overview

Main Functions:

- Login/register
- Test
- Q/A
- View Scores
- View Resources

Databases:

- Jotform
- Airtable

Integrations:

- Sendgrid
- QuickChart

Overview

Main Functions:

- Login/register
- Test
- Q/A
- View Scores
- View Resources

Databases:

- Jotform
- Airtable

Integrations:

- Sendgrid
- **QuickChart**

Databases

Jotform Tables

Questions

- Questions – questions on the Pridar standard
- Subject – broad subject of the question (such as “Design”)
- Topic – specific topic of the question (such as “User KPIs”)
- Explanation – optional explanation/context for difficult questions

Resources

- Topic – specific topic within the Pridar standard (must match above)
- Explanation – general explanation/context of the topic
- Resources – useful links to help users research the topic

QA

- Topic – concise issue or subject matter of the question
- Question – user’s question
- User – user’s email
- Answer – space to record response (needs to be emailed to user when complete)

Notes:

Changing column names doesn’t affect their question ID.
Always double check with the form itself, or a GET request.

Deleted submissions may still show up in API requests, but with their “status” field as “DELETED”.
Check and filter these results out when reading Jotform API responses.

Jotform Tables

Questions

- Questions – questions on the Pridar standard
- Subject – broad subject of the question (such as “Design”)
- Topic – specific topic of the question (such as “User KPIs”)
- Explanation – optional explanation/context for difficult questions

Resources

- Topic – specific topic within the Pridar standard (must match above)
- Explanation – general explanation/context of the topic
- Resources – useful links to help users research the topic

QA

- Topic – concise issue or subject matter of the question
- Question – user’s question
- User – user’s email
- Answer – space to record response (needs to be emailed to user when complete)

Notes:

Changing column names doesn’t affect their question ID.

Always double check with the form itself, or a GET request.

Deleted submissions may still show up in API requests, but with their “status” field as “DELETED”. Check and filter these results out when reading Jotform API responses.

Jotform Tables

Questions

- Questions – questions on the Pridar standard
- Subject – broad subject of the question (such as “Design”)
- Topic – specific topic of the question (such as “User KPIs”)
- Explanation – optional explanation/context for difficult questions

Resources

- Topic – specific topic within the Pridar standard (must match above)
- Explanation – general explanation/context of the topic
- Resources – useful links to help users research the topic

QA

- Topic – concise issue or subject matter of the question
- Question – user’s question
- User – user’s email
- Answer – space to record response (needs to be emailed to user when complete)

Notes:

Changing column names doesn’t affect their question ID.

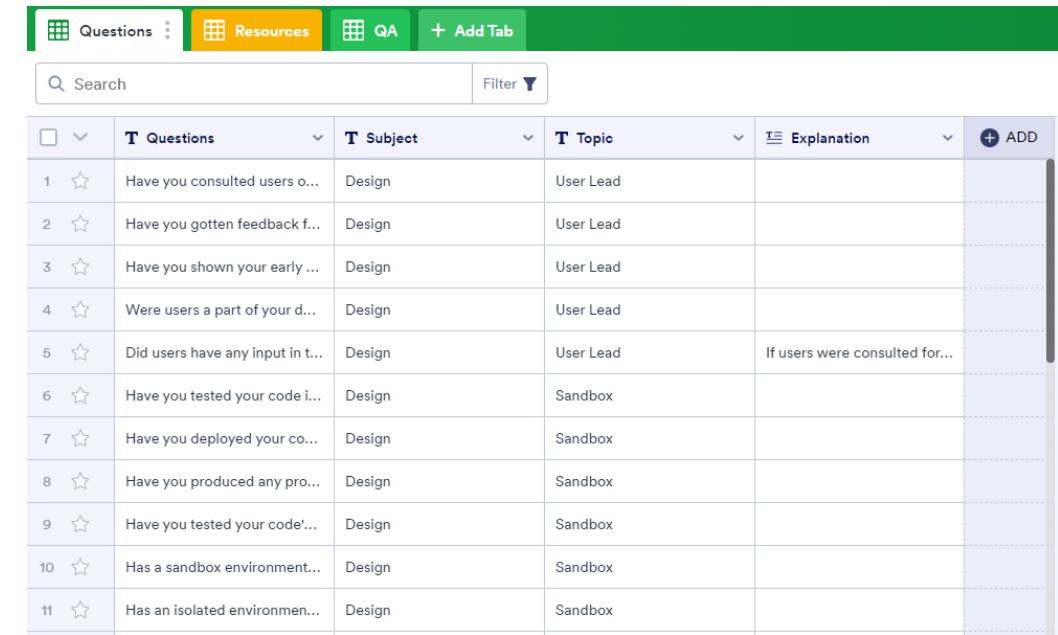
Always double check with the form itself, or a GET request.

Deleted submissions may still show up in API requests, but with their “status” field as “DELETED”. Check and filter these results out when reading Jotform API responses.

Jotform Tables

Questions

- Questions – questions on the Pridar standard
- Subject – broad subject of the question (such as “Design”)
- Topic – specific topic of the question (such as “User KPIs”)
- Explanation – optional explanation/context for difficult questions



A screenshot of a Jotform table interface. The top navigation bar includes tabs for 'Questions' (selected), 'Resources', 'QA', and '+ Add Tab'. Below the navigation is a search bar with a 'Filter' dropdown. The main table has columns: 'T Questions', 'T Subject', 'T Topic', 'T Explanation', and an 'ADD' button. The table contains 11 rows of data, each with a star icon and a question text. The 'Topic' and 'Explanation' columns are visible for the first few rows.

	T Questions	T Subject	T Topic	T Explanation	ADD
1	Have you consulted users o...	Design	User Lead		
2	Have you gotten feedback f...	Design	User Lead		
3	Have you shown your early ...	Design	User Lead		
4	Were users a part of your d...	Design	User Lead		
5	Did users have any input in t...	Design	User Lead	If users were consulted for...	
6	Have you tested your code i...	Design	Sandbox		
7	Have you deployed your co...	Design	Sandbox		
8	Have you produced any pro...	Design	Sandbox		
9	Have you tested your code'	Design	Sandbox		
10	Has a sandbox environment...	Design	Sandbox		
11	Has an isolated environmen...	Design	Sandbox		

Resources

- Topic – specific topic within the Pridar standard (must match above)
- Explanation – general explanation/context of the topic
- Resources – useful links to help users research the topic

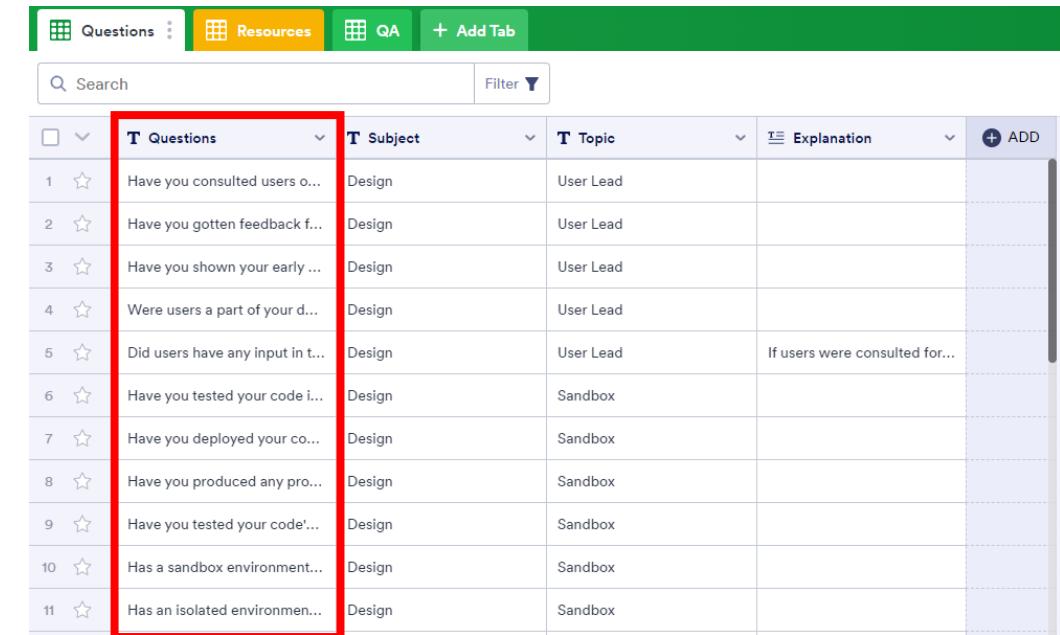
QA

- Topic – concise issue or subject matter of the question
- Question – user’s question
- User – user’s email
- Answer – space to record response (needs to be emailed to user when complete)

Jotform Tables

Questions

- **Questions – questions on the Pridar standard**
- Subject – broad subject of the question (such as “Design”)
- Topic – specific topic of the question (such as “User KPIs”)
- Explanation – optional explanation/context for difficult questions



Questions		Resources	QA	+ Add Tab
Search				
Questions	Subject	Topic	Explanation	ADD
1 Have you consulted users o...	Design	User Lead		
2 Have you gotten feedback f...	Design	User Lead		
3 Have you shown your early ...	Design	User Lead		
4 Were users a part of your d...	Design	User Lead		
5 Did users have any input in t...	Design	User Lead	If users were consulted for...	
6 Have you tested your code i...	Design	Sandbox		
7 Have you deployed your co...	Design	Sandbox		
8 Have you produced any pro...	Design	Sandbox		
9 Have you tested your code'	Design	Sandbox		
10 Has a sandbox environment...	Design	Sandbox		
11 Has an isolated environmen...	Design	Sandbox		

Resources

- Topic – specific topic within the Pridar standard (must match above)
- Explanation – general explanation/context of the topic
- Resources – useful links to help users research the topic

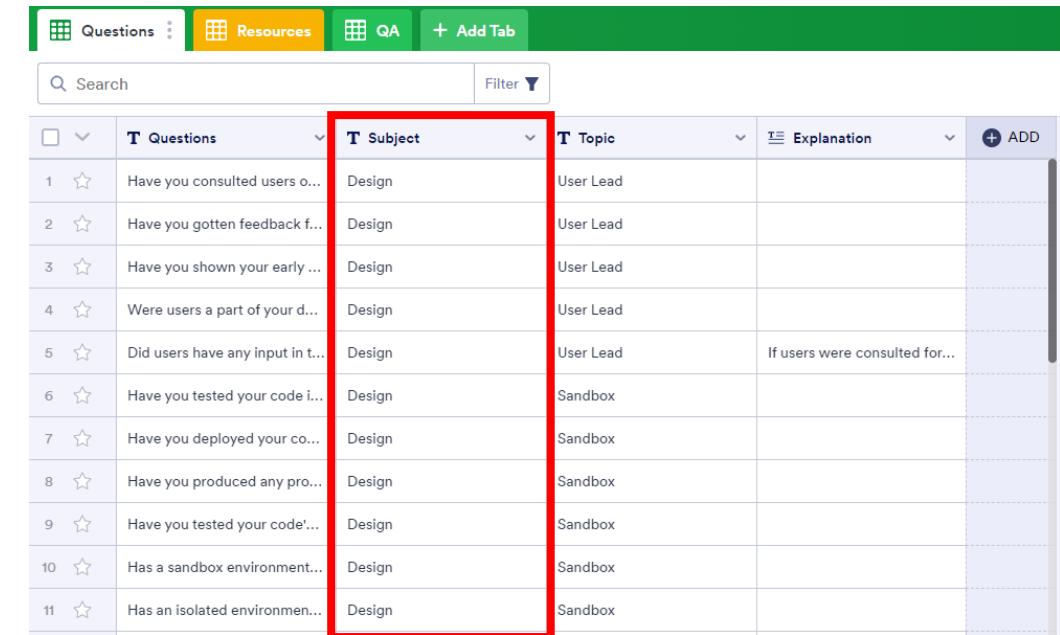
QA

- Topic – concise issue or subject matter of the question
- Question – user’s question
- User – user’s email
- Answer – space to record response (needs to be emailed to user when complete)

Jotform Tables

Questions

- Questions – questions on the Pridar standard
- **Subject – broad subject of the question (such as “Design”)**
- Topic – specific topic of the question (such as “User KPIs”)
- Explanation – optional explanation/context for difficult questions



	Questions	Resources	QA	+ Add Tab
<input type="text"/> Search Filter				
	Questions	Subject	Topic	Explanation
1	Have you consulted users o...	Design	User Lead	
2	Have you gotten feedback f...	Design	User Lead	
3	Have you shown your early ...	Design	User Lead	
4	Were users a part of your d...	Design	User Lead	
5	Did users have any input in t...	Design	User Lead	If users were consulted for...
6	Have you tested your code i...	Design	Sandbox	
7	Have you deployed your co...	Design	Sandbox	
8	Have you produced any pro...	Design	Sandbox	
9	Have you tested your code'	Design	Sandbox	
10	Has a sandbox environment...	Design	Sandbox	
11	Has an isolated environmen...	Design	Sandbox	

Resources

- Topic – specific topic within the Pridar standard (must match above)
- Explanation – general explanation/context of the topic
- Resources – useful links to help users research the topic

QA

- Topic – concise issue or subject matter of the question
- Question – user’s question
- User – user’s email
- Answer – space to record response (needs to be emailed to user when complete)

Jotform Tables

Questions

- Questions – questions on the Pridar standard
- Subject – broad subject of the question (such as “Design”)
- **Topic – specific topic of the question (such as “User KPIs”)**
- Explanation – optional explanation/context for difficult questions

	Questions	Subject	Topic	Explanation
1	Have you consulted users o...	Design	User Lead	
2	Have you gotten feedback f...	Design	User Lead	
3	Have you shown your early ...	Design	User Lead	
4	Were users a part of your d...	Design	User Lead	
5	Did users have any input in t...	Design	User Lead	If users were consulted for...
6	Have you tested your code i...	Design	Sandbox	
7	Have you deployed your co...	Design	Sandbox	
8	Have you produced any pro...	Design	Sandbox	
9	Have you tested your code'...	Design	Sandbox	
10	Has a sandbox environment...	Design	Sandbox	
11	Has an isolated environmen...	Design	Sandbox	

Resources

- Topic – specific topic within the Pridar standard (must match above)
- Explanation – general explanation/context of the topic
- Resources – useful links to help users research the topic

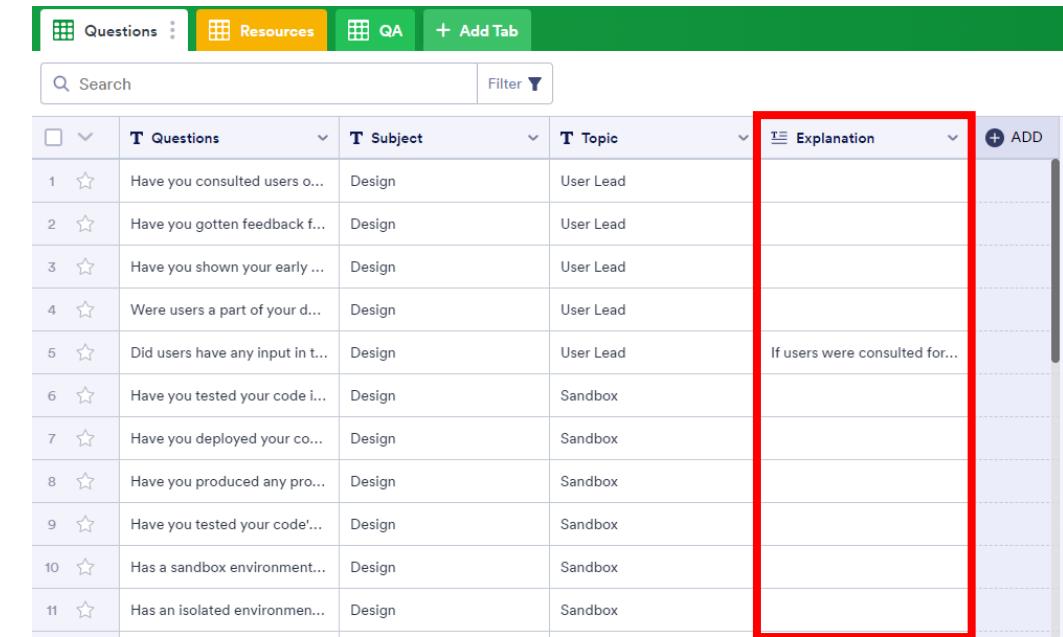
QA

- Topic – concise issue or subject matter of the question
- Question – user’s question
- User – user’s email
- Answer – space to record response (needs to be emailed to user when complete)

Jotform Tables

Questions

- Questions – questions on the Pridar standard
- Subject – broad subject of the question (such as “Design”)
- Topic – specific topic of the question (such as “User KPIs”)
- **Explanation – optional explanation/context for difficult questions**



	Questions	Resources	QA	+ Add Tab
Search				Filter
	Questions	Subject	Topic	Explanation
1	Have you consulted users o...	Design	User Lead	
2	Have you gotten feedback f...	Design	User Lead	
3	Have you shown your early ...	Design	User Lead	
4	Were users a part of your d...	Design	User Lead	
5	Did users have any input in t...	Design	User Lead	If users were consulted for...
6	Have you tested your code i...	Design	Sandbox	
7	Have you deployed your co...	Design	Sandbox	
8	Have you produced any pro...	Design	Sandbox	
9	Have you tested your code'...	Design	Sandbox	
10	Has a sandbox environment...	Design	Sandbox	
11	Has an isolated environmen...	Design	Sandbox	

Resources

- Topic – specific topic within the Pridar standard (must match above)
- Explanation – general explanation/context of the topic
- Resources – useful links to help users research the topic

QA

- Topic – concise issue or subject matter of the question
- Question – user’s question
- User – user’s email
- Answer – space to record response (needs to be emailed to user when complete)

Jotform Tables

Questions

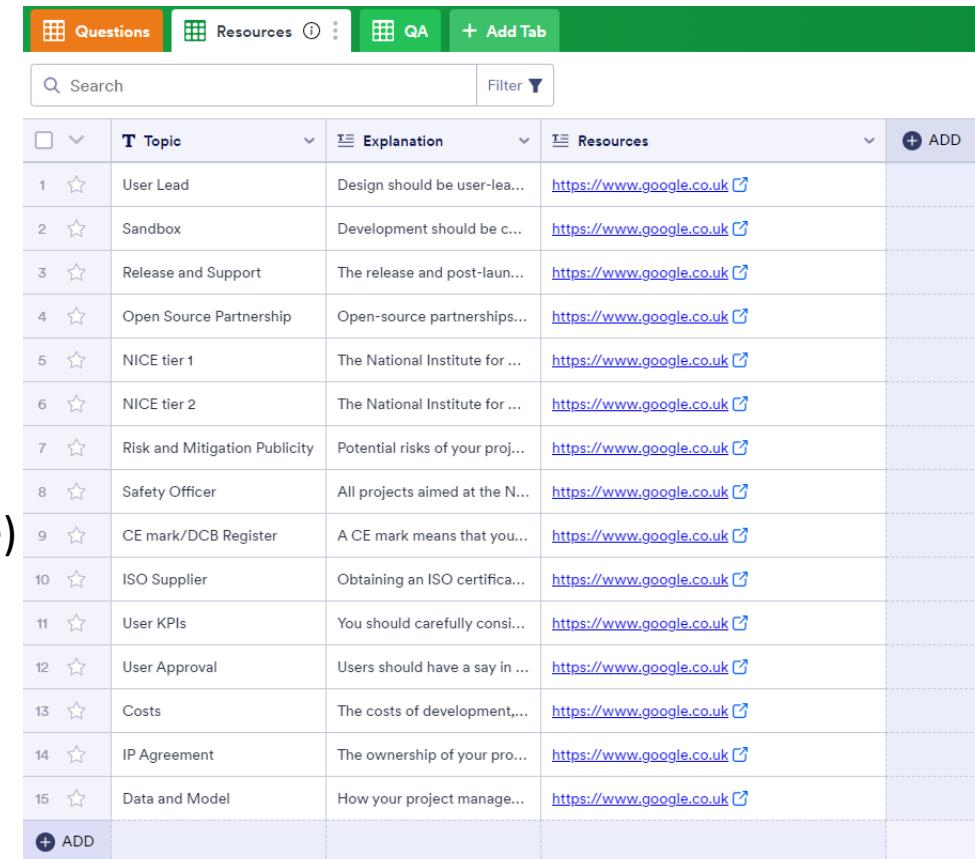
- Questions – questions on the Pridar standard
- Subject – broad subject of the question (such as “Design”)
- Topic – specific topic of the question (such as “User KPIs”)
- Explanation – optional explanation/context for difficult questions

Resources

- Topic – specific topic within the Pridar standard (must match above)
- Explanation – general explanation/context of the topic
- Resources – useful links to help users research the topic

QA

- Topic – concise issue or subject matter of the question
- Question – user’s question
- User – user’s email
- Answer – space to record response (needs to be emailed to user when complete)



The screenshot shows a Jotform Tables interface with a green header bar. The header includes tabs for 'Questions' (selected), 'Resources', 'QA', and '+ Add Tab'. Below the header is a search bar and a 'Filter' button. The main content area is a table with the following data:

	Topic	Explanation	Resources
1	User Lead	Design should be user-lea...	https://www.google.co.uk
2	Sandbox	Development should be c...	https://www.google.co.uk
3	Release and Support	The release and post-laun...	https://www.google.co.uk
4	Open Source Partnership	Open-source partnerships...	https://www.google.co.uk
5	NICE tier 1	The National Institute for ...	https://www.google.co.uk
6	NICE tier 2	The National Institute for ...	https://www.google.co.uk
7	Risk and Mitigation Publicity	Potential risks of your proj...	https://www.google.co.uk
8	Safety Officer	All projects aimed at the N...	https://www.google.co.uk
9	CE mark/DCB Register	A CE mark means that you...	https://www.google.co.uk
10	ISO Supplier	Obtaining an ISO certifica...	https://www.google.co.uk
11	User KPIs	You should carefully consi...	https://www.google.co.uk
12	User Approval	Users should have a say in ...	https://www.google.co.uk
13	Costs	The costs of development...	https://www.google.co.uk
14	IP Agreement	The ownership of your pro...	https://www.google.co.uk
15	Data and Model	How your project manage...	https://www.google.co.uk

Jotform Tables

Questions

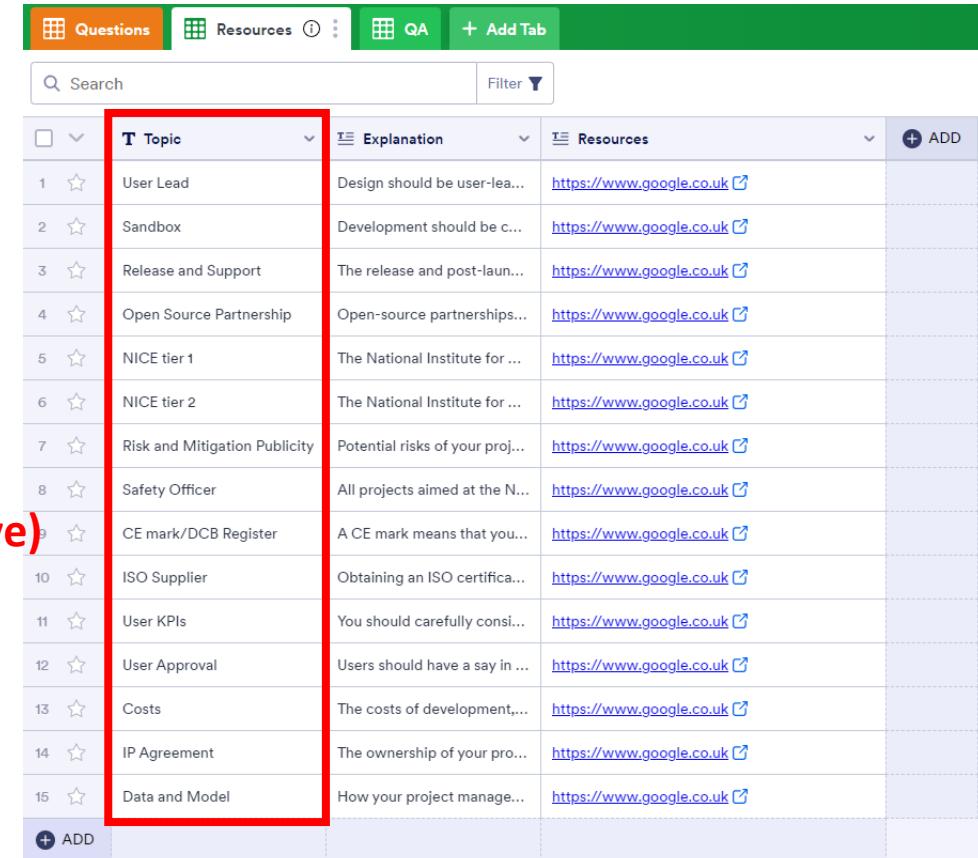
- Questions – questions on the Pridar standard
- Subject – broad subject of the question (such as “Design”)
- Topic – specific topic of the question (such as “User KPIs”)
- Explanation – optional explanation/context for difficult questions

Resources

- **Topic – specific topic within the Pridar standard (must match above)**
- Explanation – general explanation/context of the topic
- Resources – useful links to help users research the topic

QA

- Topic – concise issue or subject matter of the question
- Question – user’s question
- User – user’s email
- Answer – space to record response (needs to be emailed to user when complete)



	Topic	Explanation	Resources	
1	User Lead	Design should be user-lea...	https://www.google.co.uk	
2	Sandbox	Development should be c...	https://www.google.co.uk	
3	Release and Support	The release and post-laun...	https://www.google.co.uk	
4	Open Source Partnership	Open-source partnerships...	https://www.google.co.uk	
5	NICE tier 1	The National Institute for ...	https://www.google.co.uk	
6	NICE tier 2	The National Institute for ...	https://www.google.co.uk	
7	Risk and Mitigation Publicity	Potential risks of your proj...	https://www.google.co.uk	
8	Safety Officer	All projects aimed at the N...	https://www.google.co.uk	
9	CE mark/DCB Register	A CE mark means that you...	https://www.google.co.uk	
10	ISO Supplier	Obtaining an ISO certifica...	https://www.google.co.uk	
11	User KPIs	You should carefully consi...	https://www.google.co.uk	
12	User Approval	Users should have a say in ...	https://www.google.co.uk	
13	Costs	The costs of development...	https://www.google.co.uk	
14	IP Agreement	The ownership of your pro...	https://www.google.co.uk	
15	Data and Model	How your project manage...	https://www.google.co.uk	

Jotform Tables

Questions

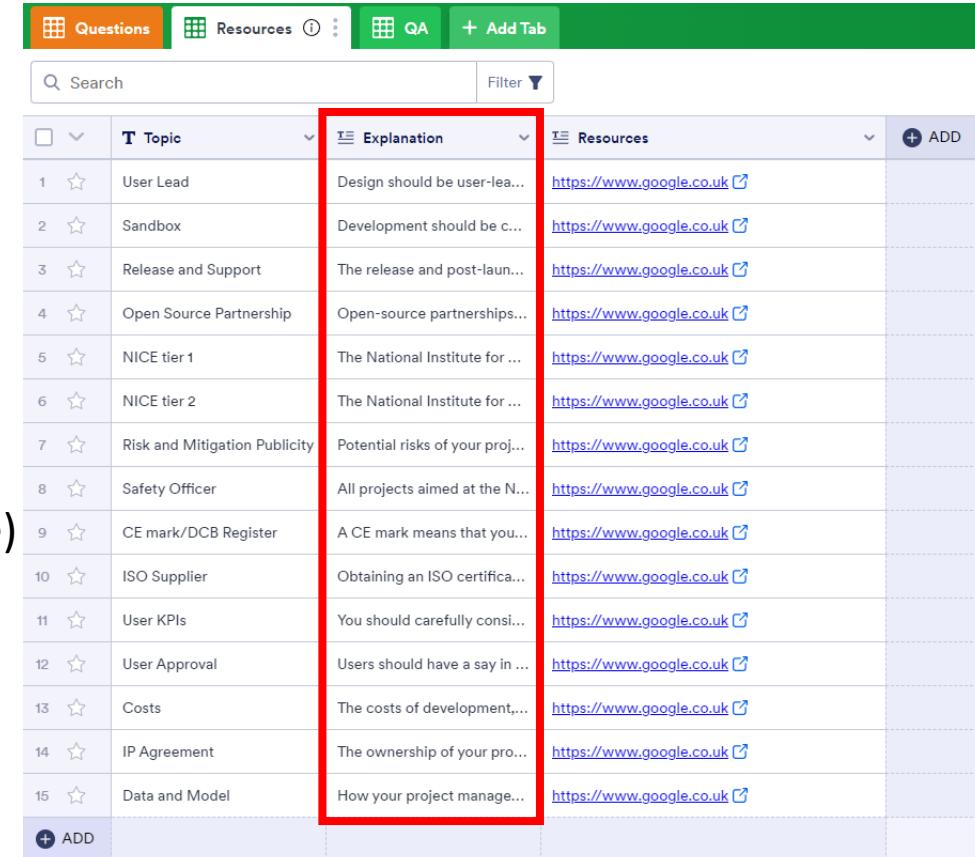
- Questions – questions on the Pridar standard
- Subject – broad subject of the question (such as “Design”)
- Topic – specific topic of the question (such as “User KPIs”)
- Explanation – optional explanation/context for difficult questions

Resources

- Topic – specific topic within the Pridar standard (must match above)
- **Explanation – general explanation/context of the topic**
- Resources – useful links to help users research the topic

QA

- Topic – concise issue or subject matter of the question
- Question – user’s question
- User – user’s email
- Answer – space to record response (needs to be emailed to user when complete)



	Topic	Explanation	Resources	ADD
1	User Lead	Design should be user-lea...	https://www.google.co.uk	
2	Sandbox	Development should be c...	https://www.google.co.uk	
3	Release and Support	The release and post-laun...	https://www.google.co.uk	
4	Open Source Partnership	Open-source partnerships...	https://www.google.co.uk	
5	NICE tier 1	The National Institute for ...	https://www.google.co.uk	
6	NICE tier 2	The National Institute for ...	https://www.google.co.uk	
7	Risk and Mitigation Publicity	Potential risks of your proj...	https://www.google.co.uk	
8	Safety Officer	All projects aimed at the N...	https://www.google.co.uk	
9	CE mark/DCB Register	A CE mark means that you...	https://www.google.co.uk	
10	ISO Supplier	Obtaining an ISO certifica...	https://www.google.co.uk	
11	User KPIs	You should carefully consi...	https://www.google.co.uk	
12	User Approval	Users should have a say in ...	https://www.google.co.uk	
13	Costs	The costs of development...	https://www.google.co.uk	
14	IP Agreement	The ownership of your pro...	https://www.google.co.uk	
15	Data and Model	How your project manage...	https://www.google.co.uk	

Jotform Tables

Questions

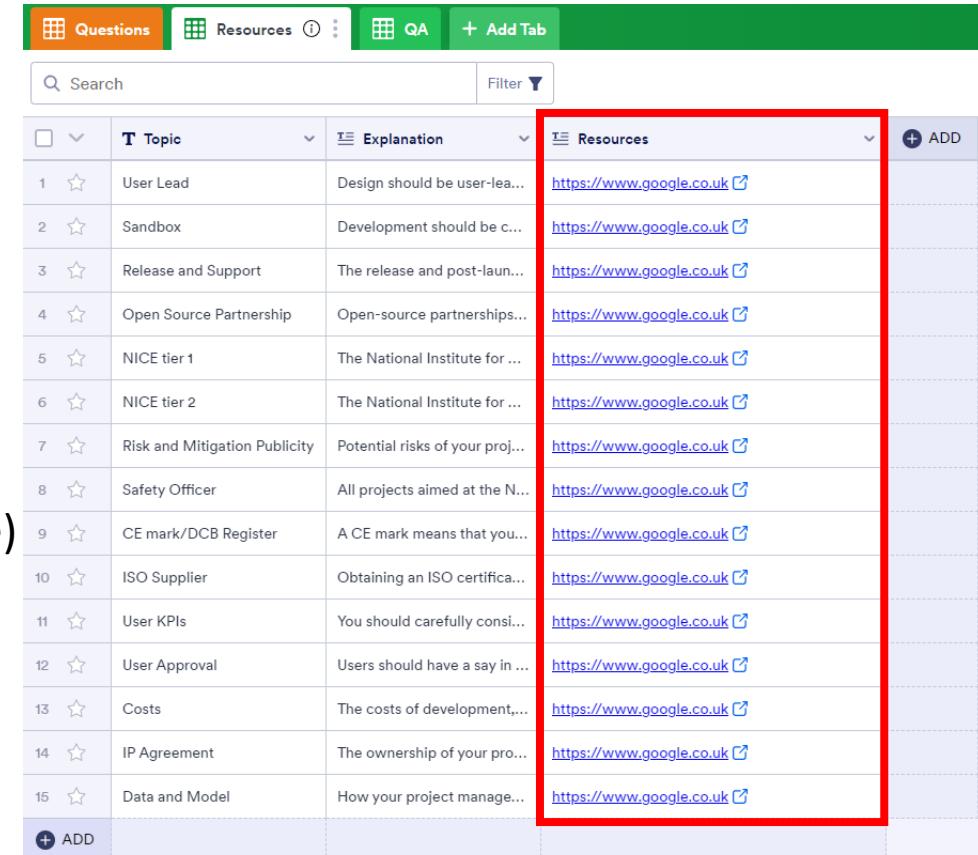
- Questions – questions on the Pridar standard
- Subject – broad subject of the question (such as “Design”)
- Topic – specific topic of the question (such as “User KPIs”)
- Explanation – optional explanation/context for difficult questions

Resources

- Topic – specific topic within the Pridar standard (must match above)
- Explanation – general explanation/context of the topic
- **Resources – useful links to help users research the topic**

QA

- Topic – concise issue or subject matter of the question
- Question – user’s question
- User – user’s email
- Answer – space to record response (needs to be emailed to user when complete)



The screenshot shows a Jotform Tables interface with a green header bar containing tabs for 'Questions', 'Resources', 'QA', and '+ Add Tab'. Below the header is a search bar and a 'Filter' dropdown. The main area is a table with the following data:

	Topic	Explanation	Resources
1	User Lead	Design should be user-lea...	https://www.google.co.uk
2	Sandbox	Development should be c...	https://www.google.co.uk
3	Release and Support	The release and post-laun...	https://www.google.co.uk
4	Open Source Partnership	Open-source partnerships...	https://www.google.co.uk
5	NICE tier 1	The National Institute for ...	https://www.google.co.uk
6	NICE tier 2	The National Institute for ...	https://www.google.co.uk
7	Risk and Mitigation Publicity	Potential risks of your proj...	https://www.google.co.uk
8	Safety Officer	All projects aimed at the N...	https://www.google.co.uk
9	CE mark/DCB Register	A CE mark means that you...	https://www.google.co.uk
10	ISO Supplier	Obtaining an ISO certifica...	https://www.google.co.uk
11	User KPIs	You should carefully consi...	https://www.google.co.uk
12	User Approval	Users should have a say in ...	https://www.google.co.uk
13	Costs	The costs of development,...	https://www.google.co.uk
14	IP Agreement	The ownership of your pro...	https://www.google.co.uk
15	Data and Model	How your project manage...	https://www.google.co.uk

Jotform Tables

Questions

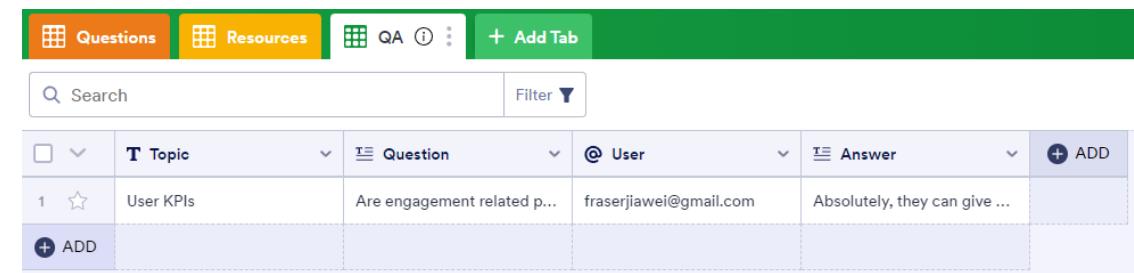
- Questions – questions on the Pridar standard
- Subject – broad subject of the question (such as “Design”)
- Topic – specific topic of the question (such as “User KPIs”)
- Explanation – optional explanation/context for difficult questions

Resources

- Topic – specific topic within the Pridar standard (must match above)
- Explanation – general explanation/context of the topic
- Resources – useful links to help users research the topic

QA

- Topic – concise issue or subject matter of the question
- Question – user’s question
- User – user’s email
- Answer – space to record response (needs to be emailed to user when complete)



	Topic	Question	User	Answer	
1	User KPIs	Are engagement related p...	fraserjiawei@gmail.com	Absolutely, they can give ...	
	+ ADD				

Jotform Tables

Questions

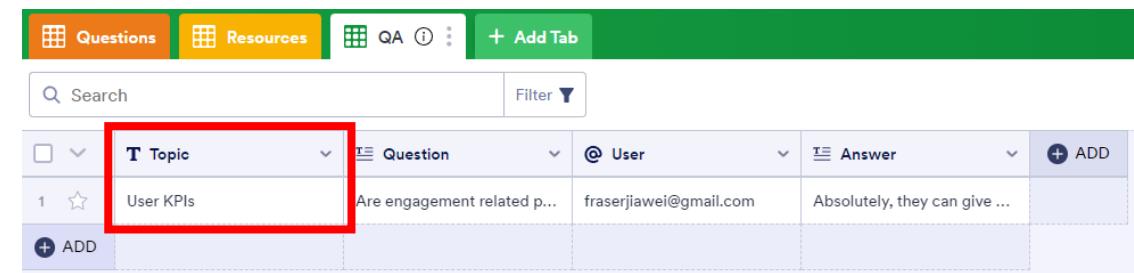
- Questions – questions on the Pridar standard
- Subject – broad subject of the question (such as “Design”)
- Topic – specific topic of the question (such as “User KPIs”)
- Explanation – optional explanation/context for difficult questions

Resources

- Topic – specific topic within the Pridar standard (must match above)
- Explanation – general explanation/context of the topic
- Resources – useful links to help users research the topic

QA

- **Topic – concise issue or subject matter of the question**
- Question – user’s question
- User – user’s email
- Answer – space to record response (needs to be emailed to user when complete)



Topic	Question	User	Answer
User KPIs	Are engagement related p...	fraserjiawei@gmail.com	Absolutely, they can give ...

Jotform Tables

Questions

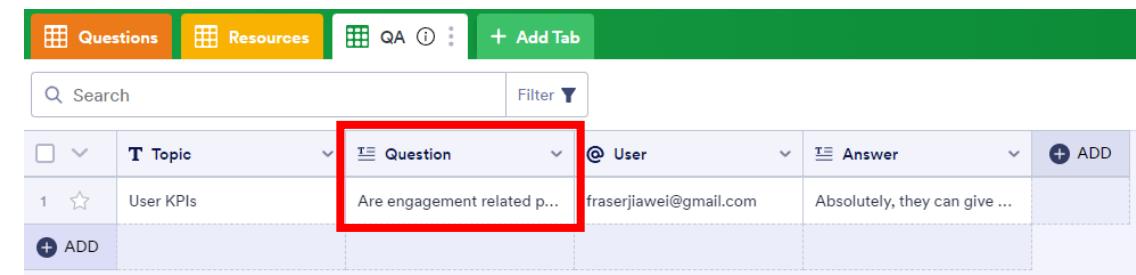
- Questions – questions on the Pridar standard
- Subject – broad subject of the question (such as “Design”)
- Topic – specific topic of the question (such as “User KPIs”)
- Explanation – optional explanation/context for difficult questions

Resources

- Topic – specific topic within the Pridar standard (must match above)
- Explanation – general explanation/context of the topic
- Resources – useful links to help users research the topic

QA

- Topic – concise issue or subject matter of the question
- **Question – user’s question**
- User – user’s email
- Answer – space to record response (needs to be emailed to user when complete)



Topic	Question	User	Answer
User KPIs	Are engagement related p...	fraserjiawei@gmail.com	Absolutely, they can give ...

Jotform Tables

Questions

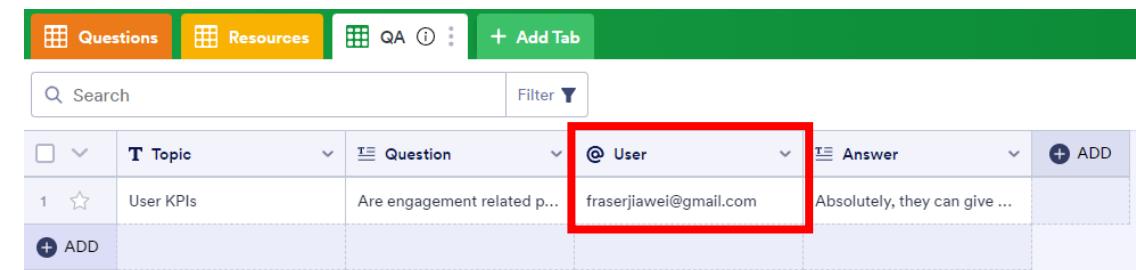
- Questions – questions on the Pridar standard
- Subject – broad subject of the question (such as “Design”)
- Topic – specific topic of the question (such as “User KPIs”)
- Explanation – optional explanation/context for difficult questions

Resources

- Topic – specific topic within the Pridar standard (must match above)
- Explanation – general explanation/context of the topic
- Resources – useful links to help users research the topic

QA

- Topic – concise issue or subject matter of the question
- Question – user’s question
- **User – user’s email**
- Answer – space to record response (needs to be emailed to user when complete)



The screenshot shows a Jotform Tables interface with a green header bar containing tabs for 'Questions', 'Resources', 'QA', and '+ Add Tab'. Below the header is a search bar and a 'Filter' button. The main area displays a table with the following data:

	Topic	Question	User	Answer	Add
1	User KPIs	Are engagement related p...	fraserjiawei@gmail.com	Absolutely, they can give ...	+ ADD
	+ ADD				

Jotform Tables

Questions

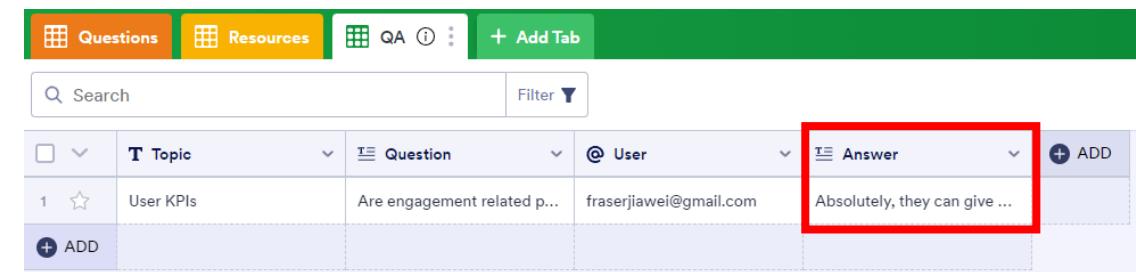
- Questions – questions on the Pridar standard
- Subject – broad subject of the question (such as “Design”)
- Topic – specific topic of the question (such as “User KPIs”)
- Explanation – optional explanation/context for difficult questions

Resources

- Topic – specific topic within the Pridar standard (must match above)
- Explanation – general explanation/context of the topic
- Resources – useful links to help users research the topic

QA

- Topic – concise issue or subject matter of the question
- Question – user’s question
- User – user’s email
- **Answer – space to record response (needs to be emailed to user when complete)**



	Topic	Question	User	Answer
1	User KPIs	Are engagement related p...	fraserjiawei@gmail.com	Absolutely, they can give ...

Airtable Tables

Credentials

- Account_Name – user's name
- Password – user's password
- Email – user's email
- Score – user's most recent scores (empty for new accounts)

Averages

- Scores – array of average scores for each topic
- Number – number of submissions in average (affects individual's weight)

Notes:

If any changes are made to the number of topics, average scores array must be updated to reflect the new size.

Remember to give the new topic an average score.

The number of submissions should be at least 50.

This will give 1% deviations in the most extreme cases (100% and 0% scores being submitted).

Airtable Tables

Credentials

- Account_Name – user's name
- Password – user's password
- Email – user's email
- Score – user's most recent scores (empty for new accounts)

Notes:

If any changes are made to the number of topics, average scores array must be updated to reflect the new size.

Remember to give the new topic an average score.

The number of submissions should be at least 50.

This will give 1% deviations in the most extreme cases (100% and 0% scores being submitted).

Averages

- Scores – array of average scores for each topic
- Number – number of submissions in average (affects individual's weight)

Airtable Tables

Credentials

- Account_Name – user's name
- Password – user's password
- Email – user's email
- Score – user's most recent scores (empty for new accounts)

Averages

- Scores – array of average scores for each topic
- Number – number of submissions in average (affects individual's weight)

Notes:

If any changes are made to the number of topics, average scores array must be updated to reflect the new size.

Remember to give the new topic an average score.

The number of submissions should be at least 50.

This will give 1% deviations in the most extreme cases (100% and 0% scores being submitted).

Airtable Tables

Credentials

- Account_Name – user's name
- Password – user's password
- Email – user's email
- Score – user's most recent scores (empty for new accounts)

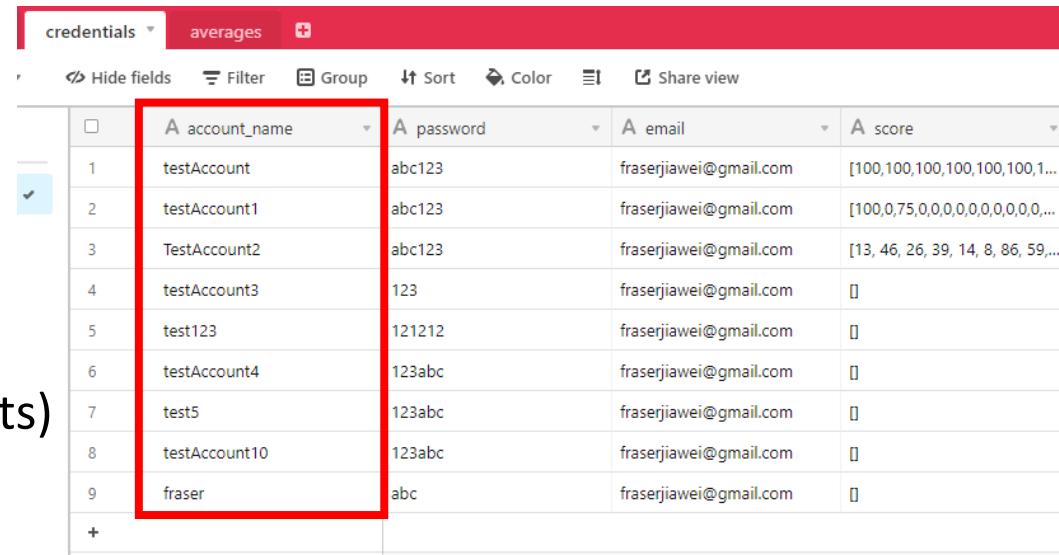
Averages

- Scores – array of average scores for each topic
- Number – number of submissions in average (affects individual's weight)

Airtable Tables

Credentials

- **Account_Name** – user's name
- Password – user's password
- Email – user's email
- Score – user's most recent scores (empty for new accounts)



	account_name	password	email	score
1	testAccount	abc123	fraserjiawei@gmail.com	[100,100,100,100,100,100,...]
2	testAccount1	abc123	fraserjiawei@gmail.com	[100,0,75,0,0,0,0,0,0,0,...]
3	TestAccount2	abc123	fraserjiawei@gmail.com	[13, 46, 26, 39, 14, 8, 86, 59,...]
4	testAccount3	123	fraserjiawei@gmail.com	
5	test123	121212	fraserjiawei@gmail.com	
6	testAccount4	123abc	fraserjiawei@gmail.com	
7	test5	123abc	fraserjiawei@gmail.com	
8	testAccount10	123abc	fraserjiawei@gmail.com	
9	fraser	abc	fraserjiawei@gmail.com	

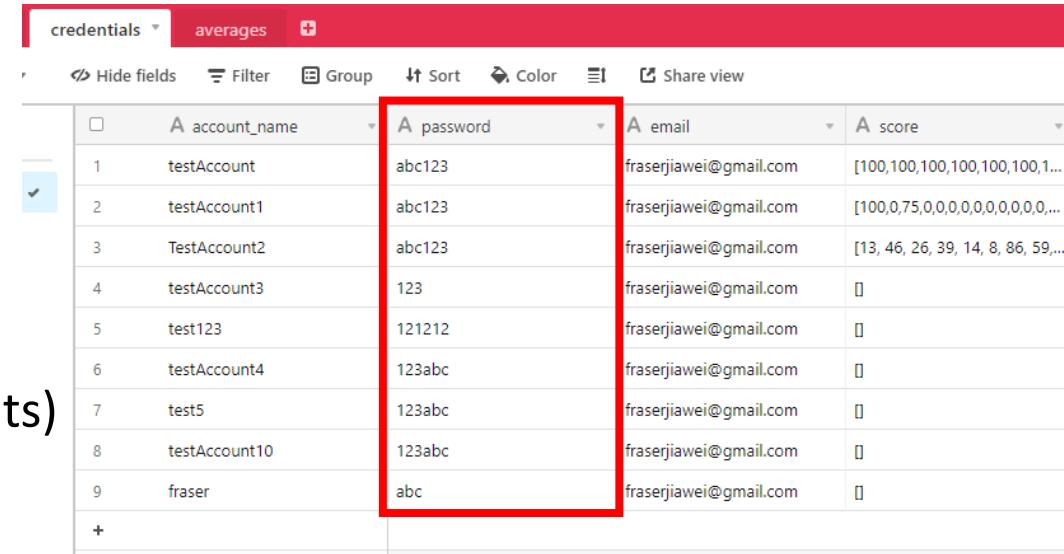
Averages

- Scores – array of average scores for each topic
- Number – number of submissions in average (affects individual's weight)

Airtable Tables

Credentials

- Account_Name – user's name
- **Password – user's password**
- Email – user's email
- Score – user's most recent scores (empty for new accounts)



The screenshot shows an Airtable table with a red header bar containing the tabs 'credentials' (selected), 'averages', and a '+' button. Below the header are standard table controls: 'Hide fields', 'Filter', 'Group', 'Sort', 'Color', 'Share view', and a 'More' icon. The table has four columns: 'account_name', 'password', 'email', and 'score'. The 'password' column is highlighted with a red box. The data rows are as follows:

	account_name	password	email	score
1	testAccount	abc123	fraserjiawei@gmail.com	[100,100,100,100,100,100,...]
2	testAccount1	abc123	fraserjiawei@gmail.com	[100,0,75,0,0,0,0,0,0,0,...]
3	TestAccount2	abc123	fraserjiawei@gmail.com	[13, 46, 26, 39, 14, 8, 86, 59,...]
4	testAccount3	123	fraserjiawei@gmail.com	□
5	test123	121212	fraserjiawei@gmail.com	□
6	testAccount4	123abc	fraserjiawei@gmail.com	□
7	test5	123abc	fraserjiawei@gmail.com	□
8	testAccount10	123abc	fraserjiawei@gmail.com	□
9	fraser	abc	fraserjiawei@gmail.com	□

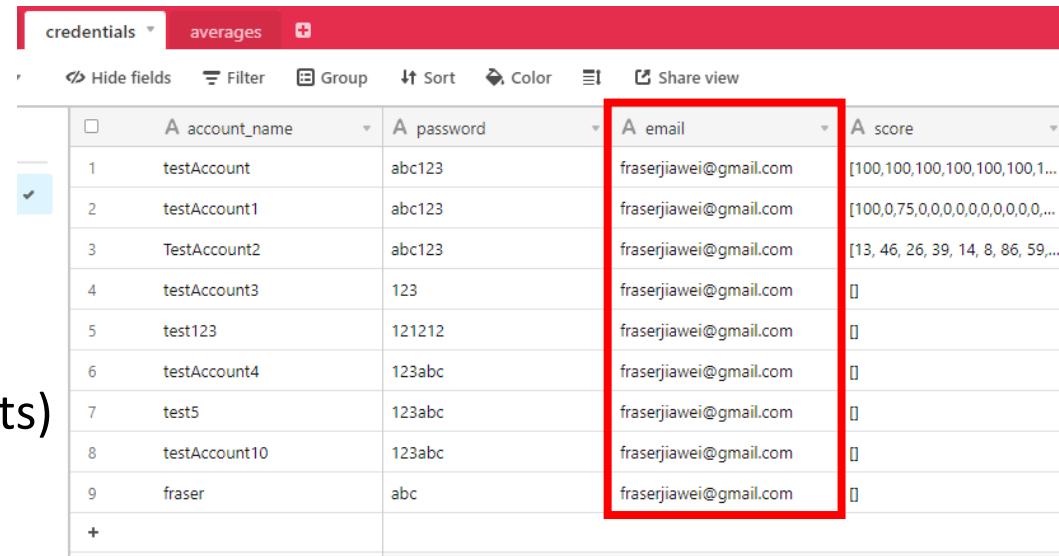
Averages

- Scores – array of average scores for each topic
- Number – number of submissions in average (affects individual's weight)

Airtable Tables

Credentials

- Account_Name – user's name
- Password – user's password
- **Email – user's email**
- Score – user's most recent scores (empty for new accounts)



The screenshot shows an Airtable table with two visible tabs: 'credentials' (selected) and 'averages'. The 'credentials' tab displays a list of accounts with columns for account name, password, and email. The 'averages' tab shows a list of users with columns for email and a 'score' field containing an array of scores. The 'email' column is highlighted with a red box.

	account_name	password	email	score
1	testAccount	abc123	fraserjiawei@gmail.com	[100,100,100,100,100,100,...]
2	testAccount1	abc123	fraserjiawei@gmail.com	[100,0,75,0,0,0,0,0,0,0,...]
3	TestAccount2	abc123	fraserjiawei@gmail.com	[13, 46, 26, 39, 14, 8, 86, 59,...]
4	testAccount3	123	fraserjiawei@gmail.com	
5	test123	121212	fraserjiawei@gmail.com	
6	testAccount4	123abc	fraserjiawei@gmail.com	
7	test5	123abc	fraserjiawei@gmail.com	
8	testAccount10	123abc	fraserjiawei@gmail.com	
9	fraser	abc	fraserjiawei@gmail.com	
+				

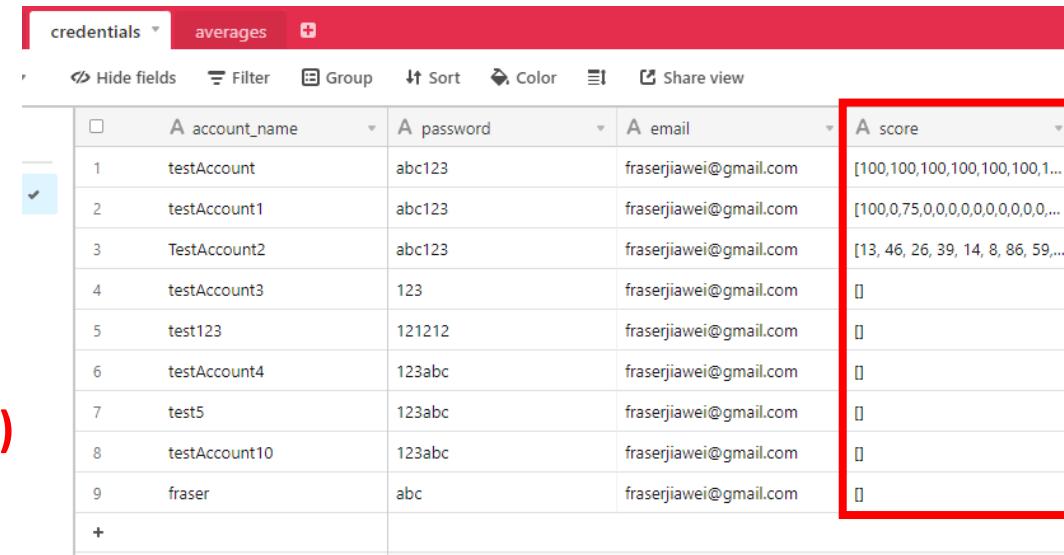
Averages

- Scores – array of average scores for each topic
- Number – number of submissions in average (affects individual's weight)

Airtable Tables

Credentials

- Account_Name – user's name
- Password – user's password
- Email – user's email
- **Score – user's most recent scores (empty for new accounts)**



The screenshot shows an Airtable table with two views: 'credentials' (selected) and 'averages'. The 'credentials' view has fields: account_name, password, email, and score. The 'averages' view is currently empty. The table contains 10 rows of data. The 'score' column is highlighted with a red box.

	account_name	password	email	score
1	testAccount	abc123	fraserjiawei@gmail.com	[100,100,100,100,100,100,...]
2	testAccount1	abc123	fraserjiawei@gmail.com	[100,0,75,0,0,0,0,0,0,0,...]
3	TestAccount2	abc123	fraserjiawei@gmail.com	[13, 46, 26, 39, 14, 8, 86, 59,...]
4	testAccount3	123	fraserjiawei@gmail.com	
5	test123	121212	fraserjiawei@gmail.com	
6	testAccount4	123abc	fraserjiawei@gmail.com	
7	test5	123abc	fraserjiawei@gmail.com	
8	testAccount10	123abc	fraserjiawei@gmail.com	
9	fraser	abc	fraserjiawei@gmail.com	
+				

Averages

- Scores – array of average scores for each topic
- Number – number of submissions in average (affects individual's weight)

Airtable Tables

Credentials

- Account_Name – user's name
- Password – user's password
- Email – user's email
- Score – user's most recent scores (empty for new accounts)

Averages

- Scores – array of average scores for each topic
- Number – number of submissions in average (affects individual's weight)

credentials	averages	+
	Hide fields	Filter
	Group	Sort
<input checked="" type="checkbox"/>	A Scores	# Number
1	[50.98039215686274, 50, 50, ...]	102
+		

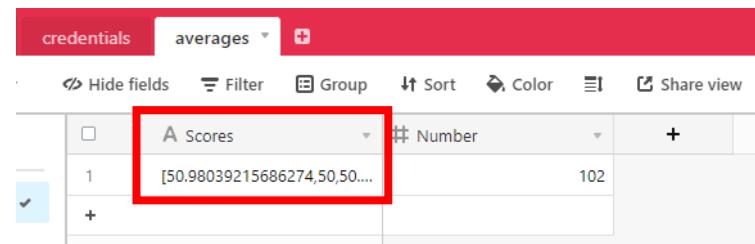
Airtable Tables

Credentials

- Account_Name – user's name
- Password – user's password
- Email – user's email
- Score – user's most recent scores (empty for new accounts)

Averages

- **Scores – array of average scores for each topic**
- Number – number of submissions in average (affects individual's weight)



	A Scores	Number
1	[50.98039215686274,50.50...]	102
+	[50.98039215686274,50.50...]	

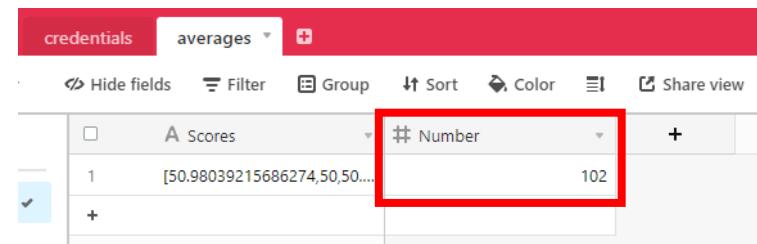
Airtable Tables

Credentials

- Account_Name – user's name
- Password – user's password
- Email – user's email
- Score – user's most recent scores (empty for new accounts)

Averages

- Scores – array of average scores for each topic
- **Number – number of submissions in average (affects individual's weight)**

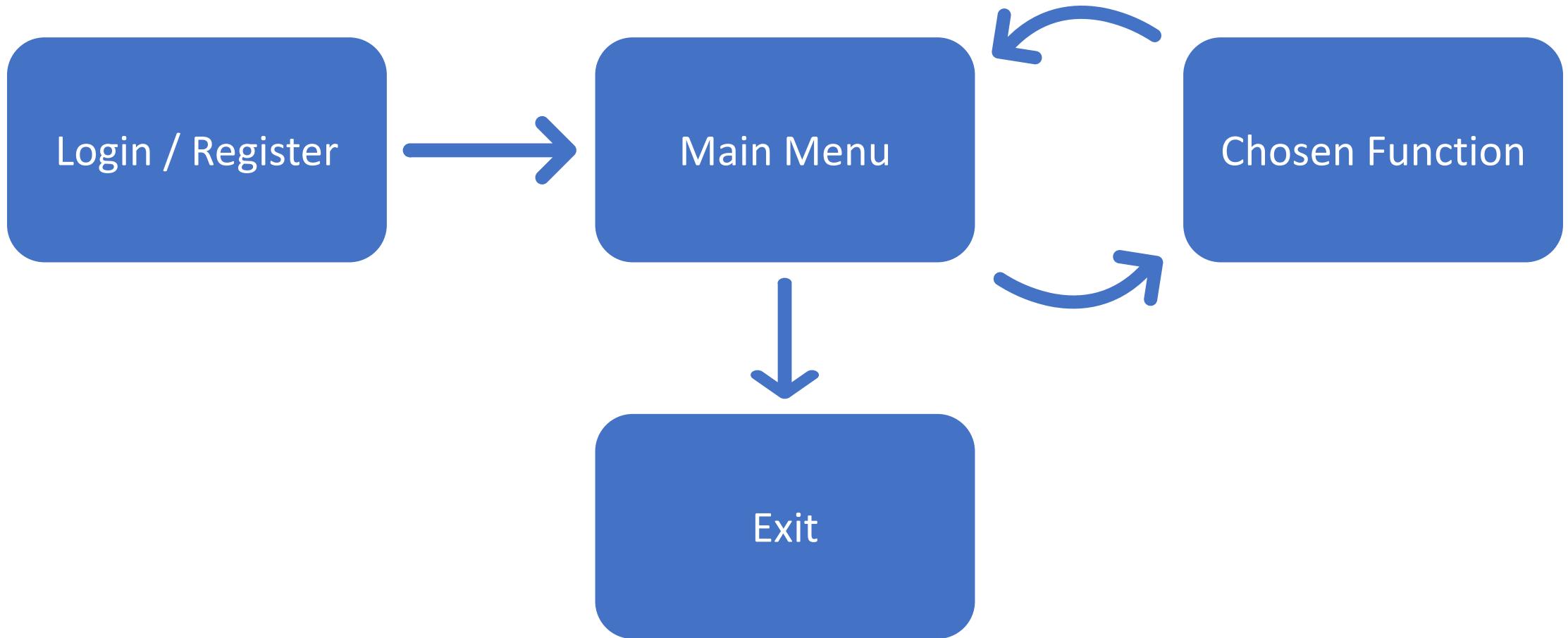


A screenshot of an Airtable table interface. The table has two columns: 'Scores' and 'Number'. The 'Scores' column contains the value '[50.98039215686274,50,50,...]'. The 'Number' column contains the value '102'. The 'Number' column is highlighted with a red box. The table has a header row with field names and a footer row with a plus sign for adding new records.

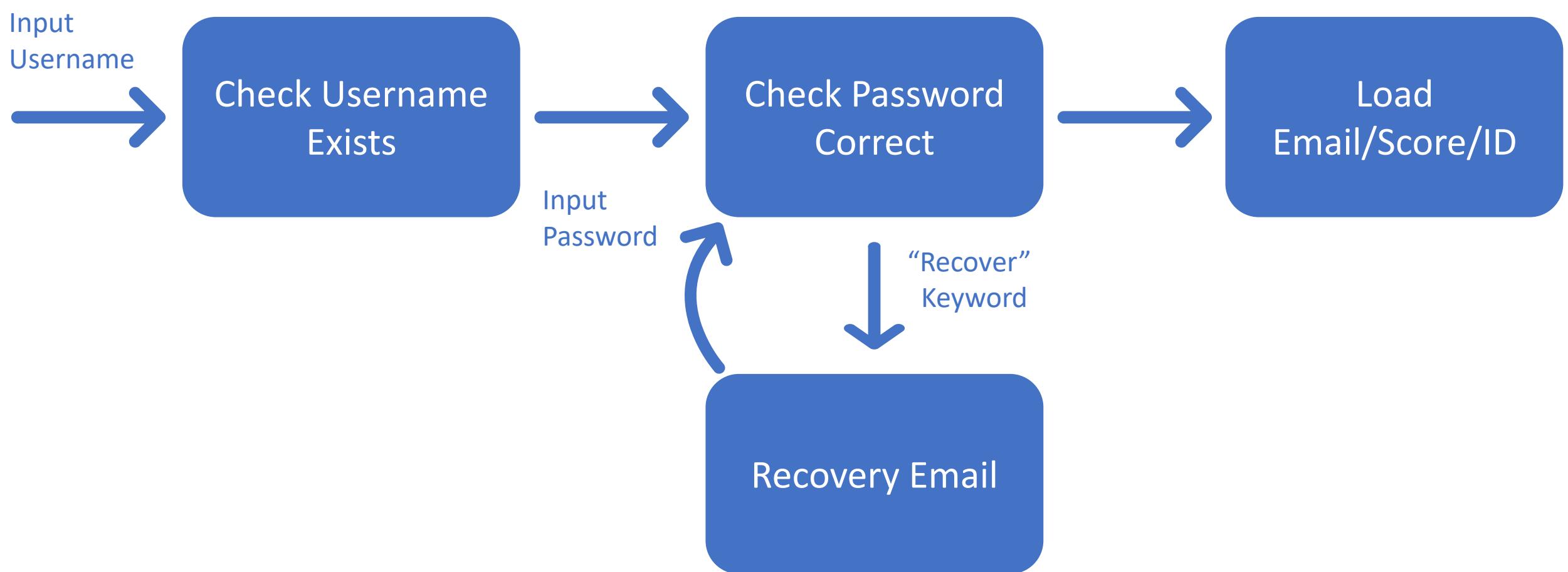
	A Scores	Number	
1	[50.98039215686274,50,50,...]	102	
+			

Algorithms

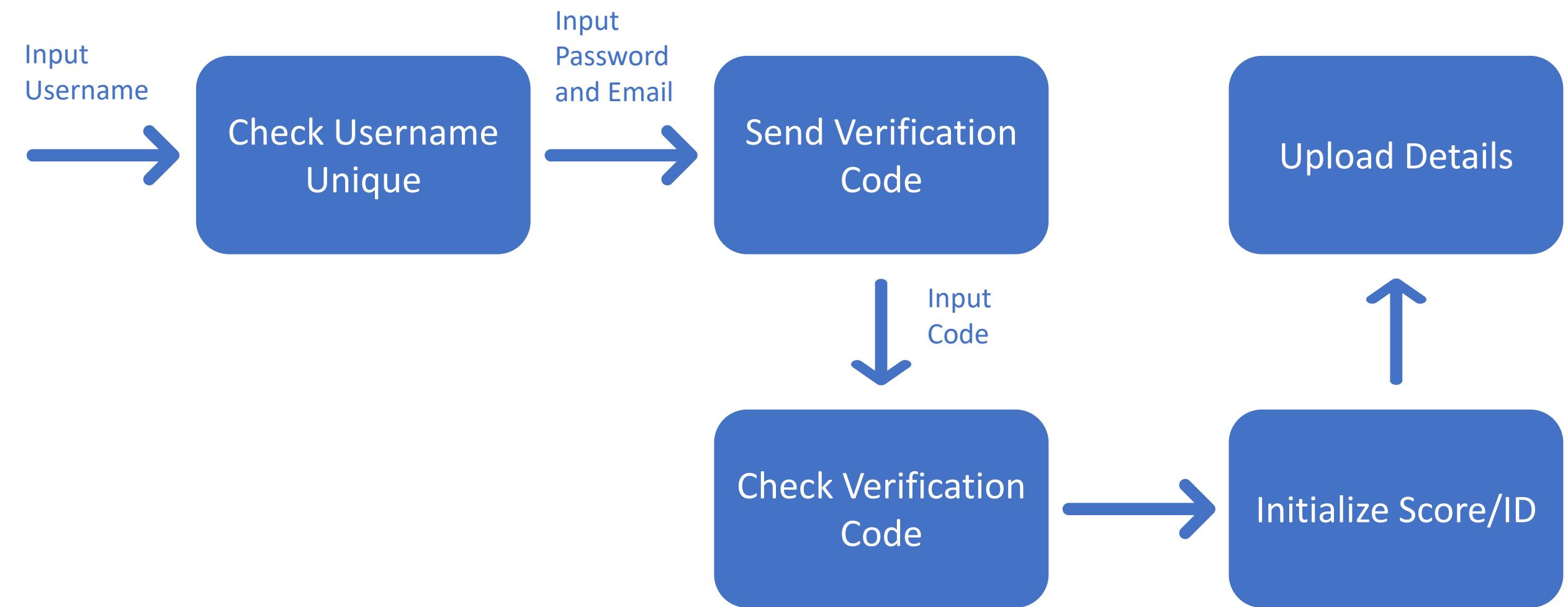
System



Login



Register



Test Overview

Retrieve/Organise data

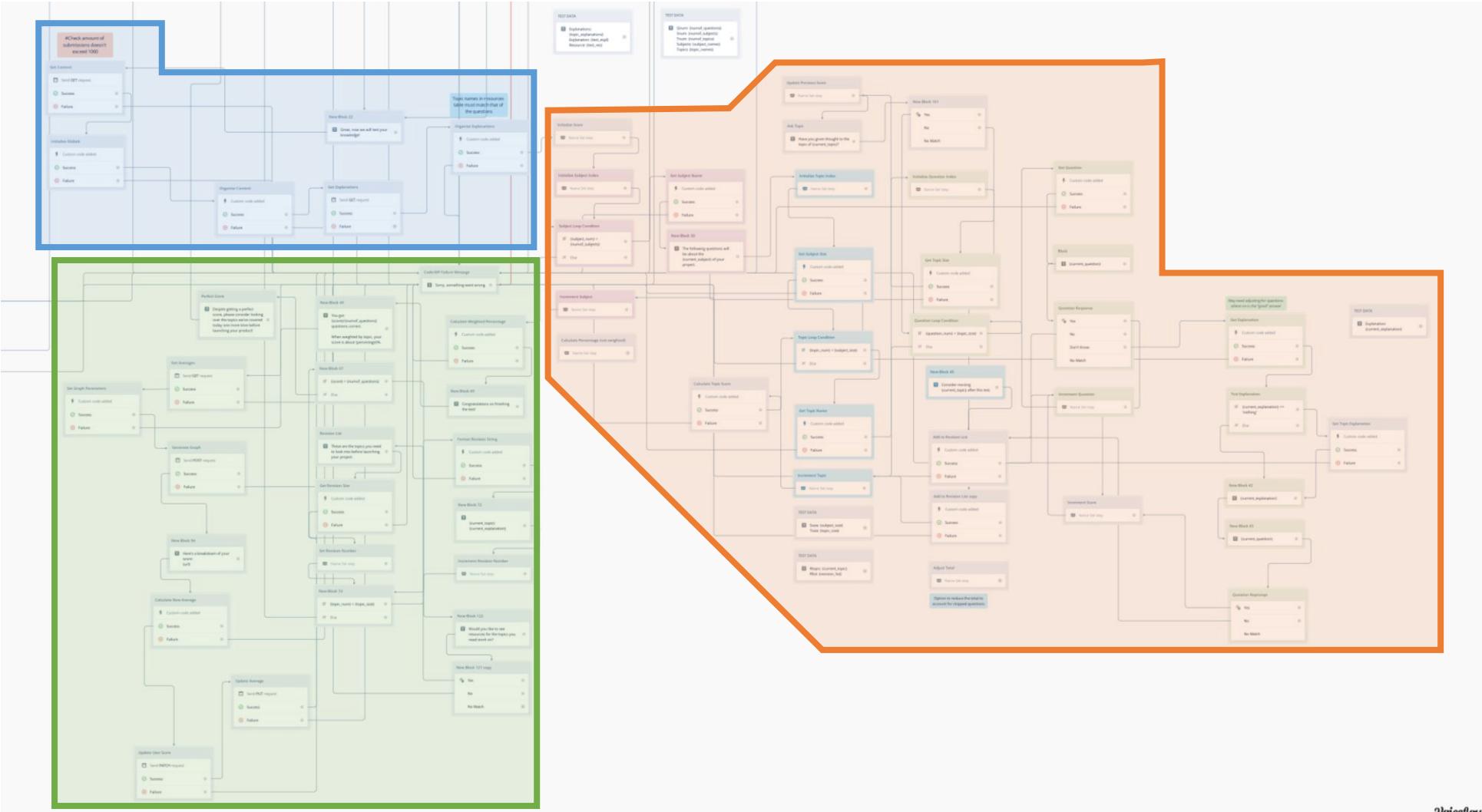


Main question loop

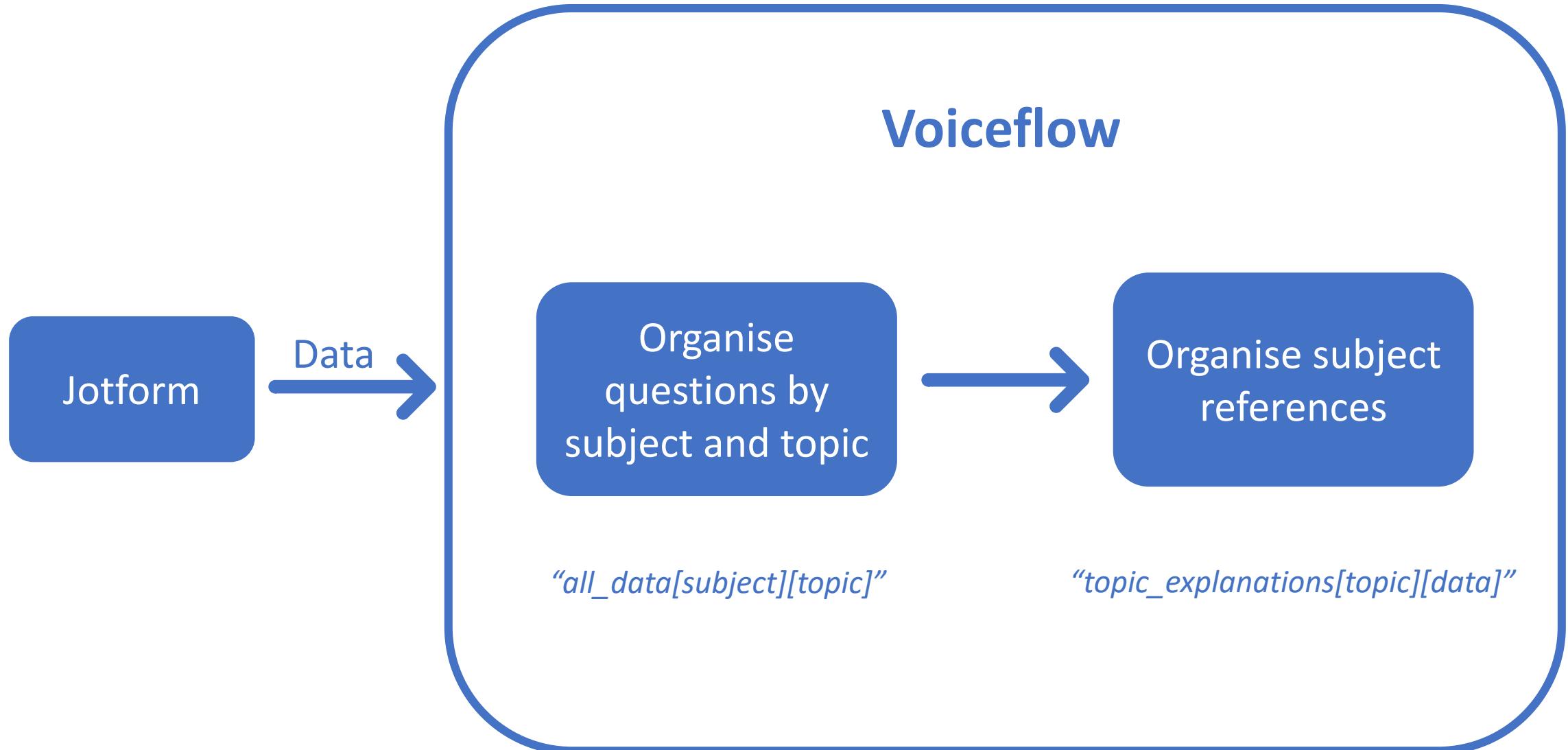


Post-test results

Test



Test Data



Question Loop

Iterate Subject

- Check subject number
- Get subject name
- Get subject size

Iterate Topic

- Check topic number
- Get topic name
- Get topic size
- Ask user to test



Topic Score = 0

YES

YES

Topic Score + 1

NO

Topic Score + 0

→

Revise Topic Later

IDK

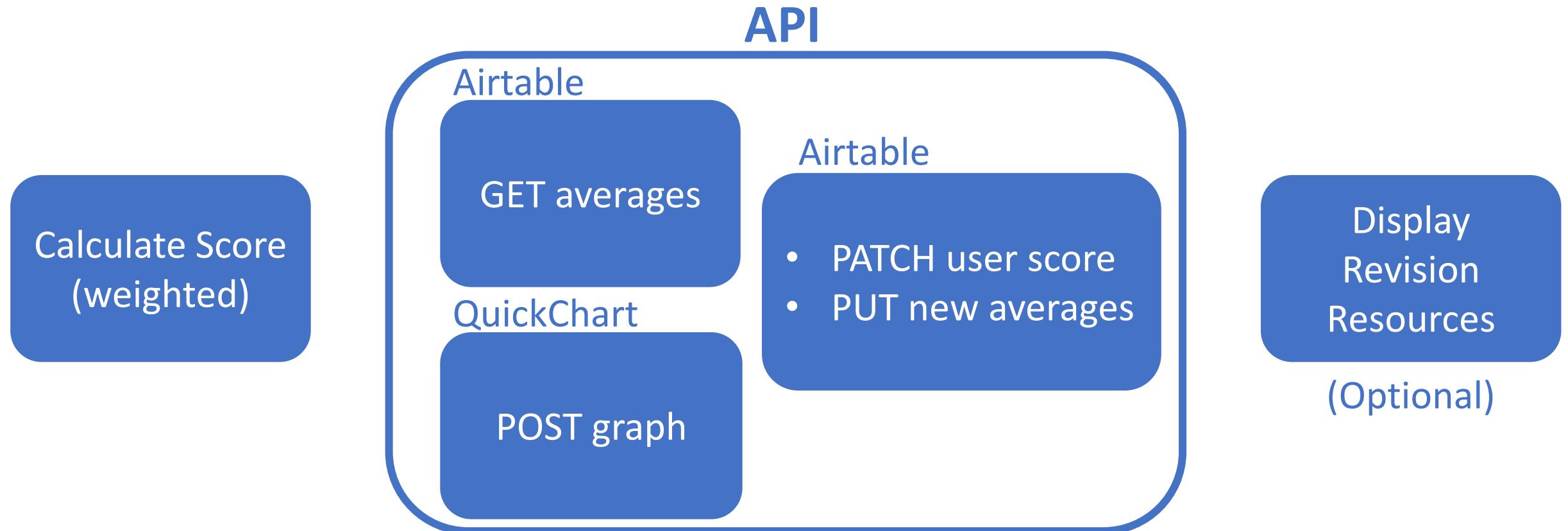
Explain

→

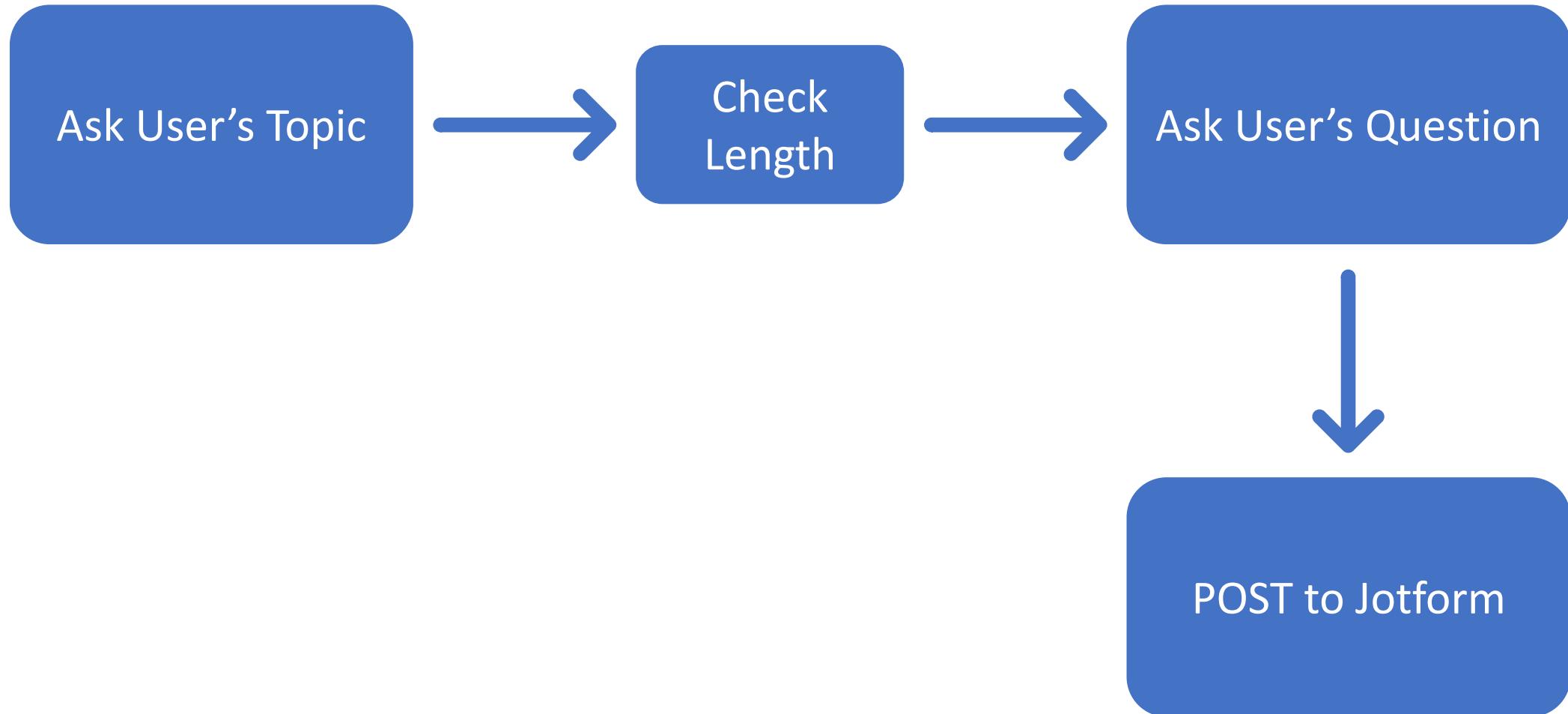
Ask Question Y/N

Ask question

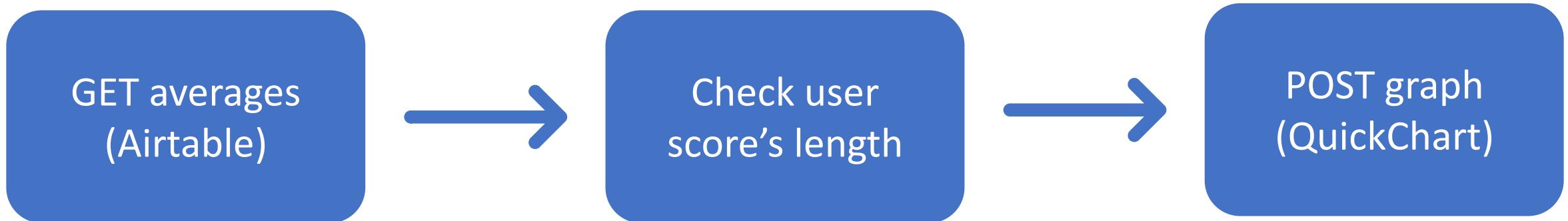
Post-Test



QA



View Scores



View Resources



“topic_information
=

[[name, explanation, resources], ...]”

Integrations

API Usage

Airtable

- Retrieve and update user account details (due to Jotform restrictions)
- Retrieve and update score averages

Sendgrid

- Email users for verification
- Email users for password recovery

Jotform

- Retrieve all information relating to the Pridar standard
- Post user queries

Quickchart

- Generate visualizations of user data

Voiceflow (via Telegram)

- Deployment of the bot

API Usage

Airtable

- **Retrieve and update user account details (due to Jotform restrictions)**
- Retrieve and update score averages

Sendgrid

- Email users for verification
- Email users for password recovery

Jotform

- Retrieve all information relating to the Pridar standard
- Post user queries

Quickchart

- Generate visualizations of user data

Voiceflow (via Telegram)

- Deployment of the bot

API Usage

Airtable

- Retrieve and update user account details (due to Jotform restrictions)
- **Retrieve and update score averages**

Sendgrid

- Email users for verification
- Email users for password recovery

Jotform

- Retrieve all information relating to the Pridar standard
- Post user queries

Quickchart

- Generate visualizations of user data

Voiceflow (via Telegram)

- Deployment of the bot

API Usage

Airtable

- Retrieve and update user account details (due to Jotform restrictions)
- Retrieve and update score averages

Sendgrid

- **Email users for verification**
- Email users for password recovery

Jotform

- Retrieve all information relating to the Pridar standard
- Post user queries

Quickchart

- Generate visualizations of user data

Voiceflow (via Telegram)

- Deployment of the bot

API Usage

Airtable

- Retrieve and update user account details (due to Jotform restrictions)
- Retrieve and update score averages

Sendgrid

- Email users for verification
- **Email users for password recovery**

Jotform

- Retrieve all information relating to the Pridar standard
- Post user queries

Quickchart

- Generate visualizations of user data

Voiceflow (via Telegram)

- Deployment of the bot

API Usage

Airtable

- Retrieve and update user account details (due to Jotform restrictions)
- Retrieve and update score averages

Sendgrid

- Email users for verification
- Email users for password recovery

Jotform

- **Retrieve all information relating to the Pridar standard**
- Post user queries

Quickchart

- Generate visualizations of user data

Voiceflow (via Telegram)

- Deployment of the bot

API Usage

Airtable

- Retrieve and update user account details (due to Jotform restrictions)
- Retrieve and update score averages

Sendgrid

- Email users for verification
- Email users for password recovery

Jotform

- Retrieve all information relating to the Pridar standard
- **Post user queries**

Quickchart

- Generate visualizations of user data

Voiceflow (via Telegram)

- Deployment of the bot

API Usage

Airtable

- Retrieve and update user account details (due to Jotform restrictions)
- Retrieve and update score averages

Sendgrid

- Email users for verification
- Email users for password recovery

Jotform

- Retrieve all information relating to the Pridar standard
- Post user queries

Quickchart

- **Generate visualizations of user data**

Voiceflow (via Telegram)

- Deployment of the bot

API Usage

Airtable

- Retrieve and update user account details (due to Jotform restrictions)
- Retrieve and update score averages

Sendgrid

- Email users for verification
- Email users for password recovery

Jotform

- Retrieve all information relating to the Pridar standard
- Post user queries

Quickchart

- Generate visualizations of user data

Voiceflow (via Telegram)

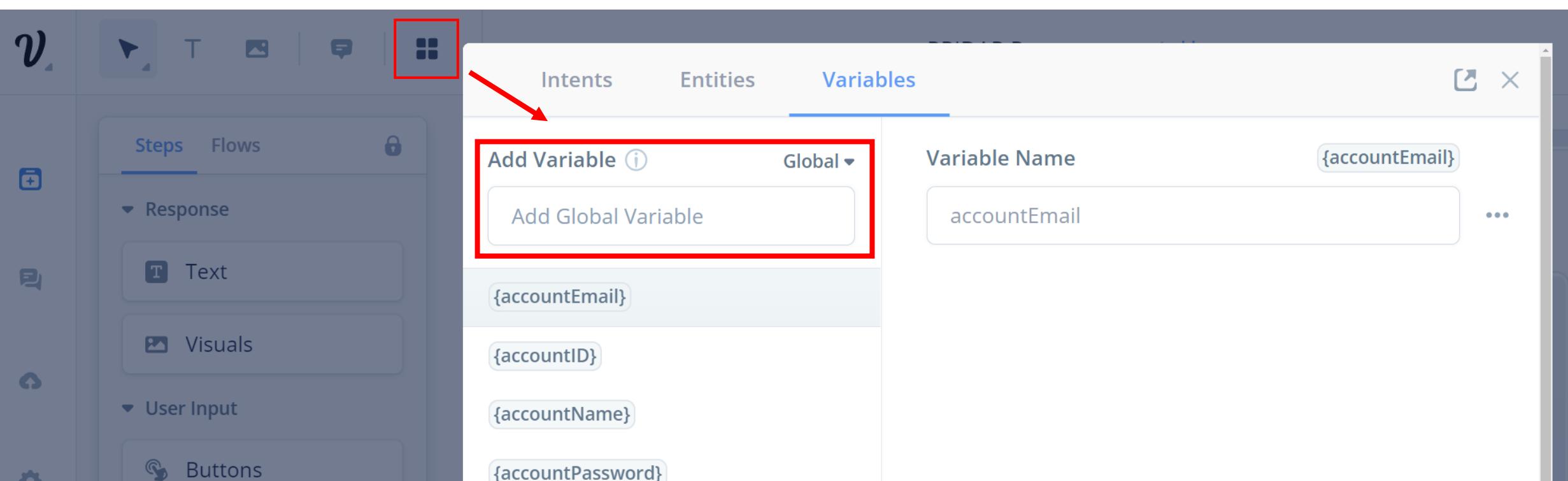
- **Deployment of the bot**

Notes

- Variables in Voiceflow must be added globally to be used outside of code blocks
- Variables in Voiceflow are accessed via `{variable_name}`, even in API bodies and parameters (the code block is the only place they may be called by name without braces)
- All API responses are in JSON format
- All data sent through these APIs should be in JSON format
- Use “`JSON.stringify(variable_name)`” to convert variables to JSON format
- Use “`JSON.parse(variable_name)`” to parse JSON into appropriate Javascript types

Notes

- **Variables in Voiceflow must be added globally to be used outside of code blocks**



Notes

- Variables in Voiceflow must be added globally to be used outside of code blocks
- **Variables in Voiceflow are accessed via {variable_name}, even in API bodies and parameters (the code block is the only place they may be called by name without braces)**
- All API responses are in JSON format
- All data sent through these APIs should be in JSON format
- Use “JSON.stringify(variable_name)” to convert variables to JSON format
- Use “JSON.parse(variable_name)” to parse JSON into appropriate Javascript types

Notes

- Variables in Voiceflow must be added globally to be used outside of code blocks
- Variables in Voiceflow are accessed via `{variable_name}`, even in API bodies and parameters (the code block is the only place they may be called by name without braces)
- **All API responses are in JSON format**
- All data sent through these APIs should be in JSON format
- Use “`JSON.stringify(variable_name)`” to convert variables to JSON format
- Use “`JSON.parse(variable_name)`” to parse JSON into appropriate Javascript types

Notes

- Variables in Voiceflow must be added globally to be used outside of code blocks
- Variables in Voiceflow are accessed via `{variable_name}`, even in API bodies and parameters (the code block is the only place they may be called by name without braces)
- All API responses are in JSON format
- **All data sent through these APIs should be in JSON format**
- Use “`JSON.stringify(variable_name)`” to convert variables to JSON format
- Use “`JSON.parse(variable_name)`” to parse JSON into appropriate Javascript types

Notes

- Variables in Voiceflow must be added globally to be used outside of code blocks
- Variables in Voiceflow are accessed via `{variable_name}`, even in API bodies and parameters (the code block is the only place they may be called by name without braces)
- All API responses are in JSON format
- All data sent through these APIs should be in JSON format
- **Use “`JSON.stringify(variable_name)`” to convert variables to JSON format**
- Use “`JSON.parse(variable_name)`” to parse JSON into appropriate Javascript types

Notes

- Variables in Voiceflow must be added globally to be used outside of code blocks
- Variables in Voiceflow are accessed via `{variable_name}`, even in API bodies and parameters (the code block is the only place they may be called by name without braces)
- All API responses are in JSON format
- All data sent through these APIs should be in JSON format
- Use “`JSON.stringify(variable_name)`” to convert variables to JSON format
- **Use “`JSON.parse(variable_name)`” to parse JSON into appropriate Javascript types**

Jotform API

GET Request

- Add form ID in request url and specify “/submissions”
- API key to authorise requests
- Store response in variable
- Set parameter “limit” to 1000
- Specify columns by question ID (obtained in Jotform)
“request.content[x].answers[n].answer”

Request URL

GET ▾ rm.com/form/220372763514049/submissions

Headers Params

Header Assignments

APIKEY

VALUE `f126742005e285e3de986aed18b669ef`

Transform into Variables

response

APPLY TO request

Request URL

GET ▾ <https://eu-api.jotform.com/form/2203727635>

Headers Params

Parameter Assignments

limit

VALUE 1000

Response:

```
{
  "responseCode": 200,
  "message": "success",
  "content": [
    {
      "id": "5202970564526659949",
      "form_id": "220372763514049",
      "ip": "92.0.35.254",
      "created_at": "2022-02-10 05:10:56",
      "status": "CUSTOM",
      "new": "0",
      "flag": "0",
      "notes": "",
      "updated_at": "2022-02-16 06:00:03",
      "answers": [
        "1": {
          "name": "heading",
          "order": "1",
          "text": "Resources",
          "type": "control_head"
        },
        "2": {
          "hackRecipients": "No",
          "name": "submit2",
          "order": "5",
          "text": "Submit",
          "type": "control_button"
        },
        "3": {
          "name": "name",
          "order": "2",
          "text": "Topic",
          "type": "control_textbox",
          "answer": "Data and Model"
        },
        "4": {
          "name": "explanation",
          "order": "3",
          "text": "Explanation",
          "type": "control_textarea",
          "answer": "How your project manages data should be planned in detail, considering scale and"
        },
        "5": {
          "name": "resources",
          "order": "4",
          "text": "Resources",
          "type": "control_textarea",
          "answer": "https://www.google.co.uk"
        }
      ]
    }
  ]
}
```

GET Request

- **Add form ID in request url and specify “/submissions”**
- API key to authorise requests
- Store response in variable
- Set parameter “limit” to 1000
- Specify columns by question ID (obtained in Jotform)
“request.content[x].answers[n].answer”

Request URL

GET ▾ rm.com/form/220372763514049/submissions

Headers Params

Header Assignments

+ APIKEY

- VALUE f126742005e285e3de986aed18b669ef

Transform into Variables

+ response

- APPLY TO request

Request URL

GET ▾ <https://eu-api.jotform.com/form/2203727635>

Headers Params

Parameter Assignments

+ limit

- VALUE 1000

Response:

```
{
  "responseCode": 200,
  "message": "success",
  "content": [
    {
      "id": "5202970564526659949",
      "form_id": "220372763514049",
      "ip": "92.0.35.254",
      "created_at": "2022-02-10 05:10:56",
      "status": "CUSTOM",
      "new": "0",
      "flag": "0",
      "notes": "",
      "updated_at": "2022-02-16 06:00:03",
      "answers": [
        "1": {
          "name": "heading",
          "order": "1",
          "text": "Resources",
          "type": "control_head"
        },
        "2": {
          "hackRecipients": "No",
          "name": "submit2",
          "order": "5",
          "text": "Submit",
          "type": "control_button"
        },
        "3": {
          "name": "name",
          "order": "2",
          "text": "Topic",
          "type": "control_textbox",
          "answer": "Data and Model"
        },
        "4": {
          "name": "explanation",
          "order": "3",
          "text": "Explanation",
          "type": "control_textarea",
          "answer": "How your project manages data should be planned in detail, considering scale and"
        },
        "5": {
          "name": "resources",
          "order": "4",
          "text": "Resources",
          "type": "control_textarea",
          "answer": "https://www.google.co.uk"
        }
      ]
    }
  ]
}
```

GET Request

- Add form ID in request url and specify “/submissions”
- **API key to authorise requests**
- Store response in variable
- Set parameter “limit” to 1000
- Specify columns by question ID (obtained in Jotform)
“request.content[x].answers[n].answer”

Request URL

GET ▾ rm.com/form/220372763514049/submissions

Headers Params

Header Assignments

+ APIKEY
-
VALUE f126742005e285e3de986aed18b669ef

Transform into Variables

+ response
-
APPLY TO request

Request URL

GET ▾ <https://eu-api.jotform.com/form/2203727635>

Headers Params

Parameter Assignments

+ limit
-
VALUE 1000

Response:

```
{  
  "responseCode": 200,  
  "message": "success",  
  "content": [  
    {  
      "id": "5202970564526659949",  
      "form_id": "220372763514049",  
      "ip": "92.0.35.254",  
      "created_at": "2022-02-10 05:10:56",  
      "status": "CUSTOM",  
      "new": "0",  
      "flag": "0",  
      "notes": "",  
      "updated_at": "2022-02-16 06:00:03",  
      "answers": {  
        "1": {  
          "name": "heading",  
          "order": "1",  
          "text": "Resources",  
          "type": "control_head"  
        },  
        "2": {  
          "hackRecipients": "No",  
          "name": "submit2",  
          "order": "5",  
          "text": "Submit",  
          "type": "control_button"  
        },  
        "3": {  
          "name": "name",  
          "order": "2",  
          "text": "Topic",  
          "type": "control_textbox",  
          "answer": "Data and Model"  
        },  
        "4": {  
          "name": "explanation",  
          "order": "3",  
          "text": "Explanation",  
          "type": "control_textarea",  
          "answer": "How your project manages data should be planned in detail, considering scale and  
        },  
        "5": {  
          "name": "resources",  
          "order": "4",  
          "text": "Resources",  
          "type": "control_textarea",  
          "answer": "https://www.google.co.uk"  
        }  
      }  
    }  
  ]  
}
```

GET Request

- Add form ID in request url and specify “/submissions”
- API key to authorise requests
- **Store response in variable**
- Set parameter “limit” to 1000
- Specify columns by question ID (obtained in Jotform)
“request.content[x].answers[n].answer”

Request URL

GET ▾ rm.com/form/220372763514049/submissions

Headers Params

Header Assignments

+ APIKEY

-

VALUE f126742005e285e3de986aed18b669ef

Transform into Variables

+ response

-

APPLY TO request

Request URL

GET ▾ https://eu-api.jotform.com/form/2203727635

Headers Params

Parameter Assignments

+ limit

-

VALUE 1000

Response:

```
{  
  "responseCode": 200,  
  "message": "success",  
  "content": [  
    {  
      "id": "5202970564526659949",  
      "form_id": "220372763514049",  
      "ip": "92.0.35.254",  
      "created_at": "2022-02-10 05:10:56",  
      "status": "CUSTOM",  
      "new": "0",  
      "flag": "0",  
      "notes": "",  
      "updated_at": "2022-02-16 06:00:03",  
      "answers": [  
        "1": {  
          "name": "heading",  
          "order": "1",  
          "text": "Resources",  
          "type": "control_head"  
        },  
        "2": {  
          "hackRecipients": "No",  
          "name": "submit2",  
          "order": "5",  
          "text": "Submit",  
          "type": "control_button"  
        },  
        "3": {  
          "name": "name",  
          "order": "2",  
          "text": "Topic",  
          "type": "control_textbox",  
          "answer": "Data and Model"  
        },  
        "4": {  
          "name": "explanation",  
          "order": "3",  
          "text": "Explanation",  
          "type": "control_textarea",  
          "answer": "How your project manages data should be planned in detail, considering scale and"  
        },  
        "5": {  
          "name": "resources",  
          "order": "4",  
          "text": "Resources",  
          "type": "control_textarea",  
          "answer": "https://www.google.co.uk"  
        }  
      ]  
    }  
  ]  
}
```

GET Request

- Add form ID in request url and specify “/submissions”
- API key to authorise requests
- Store response in variable
- **Set parameter “limit” to 1000**
- Specify columns by question ID (obtained in Jotform)
“request.content[x].answers[n].answer”

Request URL

GET ▾ rm.com/form/220372763514049/submissions

Headers Params

Header Assignments

+ APIKEY

- VALUE f126742005e285e3de986aed18b669ef

Transform into Variables

+ response

- APPLY TO request

Request URL

GET ▾ <https://eu-api.jotform.com/form/2203727635>

Headers Params

Parameter Assignments

+ limit

- VALUE 1000

Response:

```
{
  "responseCode": 200,
  "message": "success",
  "content": [
    {
      "id": "5202970564526659949",
      "form_id": "220372763514049",
      "ip": "92.0.35.254",
      "created_at": "2022-02-10 05:10:56",
      "status": "CUSTOM",
      "new": "0",
      "flag": "0",
      "notes": "",
      "updated_at": "2022-02-16 06:00:03",
      "answers": [
        "1": {
          "name": "heading",
          "order": "1",
          "text": "Resources",
          "type": "control_head"
        },
        "2": {
          "hackRecipients": "No",
          "name": "submit2",
          "order": "5",
          "text": "Submit",
          "type": "control_button"
        },
        "3": {
          "name": "name",
          "order": "2",
          "text": "Topic",
          "type": "control_textbox",
          "answer": "Data and Model"
        },
        "4": {
          "name": "explanation",
          "order": "3",
          "text": "Explanation",
          "type": "control_textarea",
          "answer": "How your project manages data should be planned in detail, considering scale and"
        },
        "5": {
          "name": "resources",
          "order": "4",
          "text": "Resources",
          "type": "control_textarea",
          "answer": "https://www.google.co.uk"
        }
      ]
    }
  ]
}
```

GET Request

- Add form ID in request url and specify “/submissions”
- API key to authorise requests
- Store response in variable
- Set parameter “limit” to 1000
- **Specify columns by question ID (obtained in Jotform)**
“request.content[x].answers[n].answer”

Request URL

GET ▾ rm.com/form/220372763514049/submissions

Headers Params

Header Assignments

APIKEY

VALUE f126742005e285e3de986aed18b669ef

Transform into Variables

response

APPLY TO request ▾

Request URL

GET ▾ https://eu-api.jotform.com/form/2203727635

Headers Params

Parameter Assignments

limit

VALUE 1000

Response:

```
{  
  "responseCode": 200,  
  "message": "success",  
  "content": [  
    {  
      "id": "5202970564526659949",  
      "form_id": "220372763510409",  
      "ip": "92.0.35.254",  
      "created_at": "2022-02-10 05:10:56",  
      "status": "CUSTOM",  
      "new": "0",  
      "flag": "0",  
      "notes": "",  
      "updated_at": "2022-02-16 06:00:03",  
      "answers": {  
        "1": {  
          "name": "heading",  
          "order": "1",  
          "text": "Resources",  
          "type": "control_head"  
        },  
        "2": {  
          "hackRecipients": "No",  
          "name": "submit2",  
          "order": "5",  
          "text": "Submit",  
          "type": "control_button"  
        },  
        "3": {  
          "name": "name",  
          "order": "2",  
          "text": "Topic",  
          "type": "control_textbox",  
          "answer": "Data and Model"  
        },  
        "4": {  
          "name": "explanation",  
          "order": "3",  
          "text": "Explanation",  
          "type": "control_textarea",  
          "answer": "How your project manages data should be planned in detail, considering scale and  
        },  
        "5": {  
          "name": "resources",  
          "order": "4",  
          "text": "Resources",  
          "type": "control_textarea",  
          "answer": "https://www.google.co.uk"  
        },  
      }  
    }  
  ]  
}
```

POST Request

- Add form ID in request url and specify “/submissions”
- API key to authorise requests
- Specify columns by question ID (obtained in Jotform)
“submission[n]”
- Add parameter for each column submission

Request URL

POST ▾ https://eu-api.jotform.com/form/220332334

Headers Body Params (3)

Parameter Assignments

+	submission[11]	-
	VALUE {user_topic}	
+	submission[12]	-
	VALUE {user_question}	
+	submission[13]	-
	VALUE {accountEmail}	

Request URL

POST ▾ https://eu-api.jotform.com/form/220332334

Headers Body Params (3)

Header Assignments

+	APIKEY	-
	VALUE f126742005e285e3de986aed18b669ef	

Transform into Variables

Enter object path

APPLY TO Select Variable

POST Request

- **Add form ID in request url and specify “/submissions”**
- API key to authorise requests
- Specify columns by question ID (obtained in Jotform)
“submission[n]”
- Add parameter for each column submission

Request URL

POST ▾ <https://eu-api.jotform.com/form/220332334801038/submissions>

Headers Body Params (3)

Parameter Assignments

submission[11]	+	-
VALUE {user_topic}	+	-
submission[12]	+	-
VALUE {user_question}	+	-
submission[13]	+	-
VALUE {accountEmail}	+	-

Request URL

POST ▾ <https://eu-api.jotform.com/form/220332334>

Headers Body Params (3)

Header Assignments

APIKEY	+	-
VALUE f126742005e285e3de986aed18b669ef	+	-

Transform into Variables

Enter object path	+	-
APPLY TO Select Variable	+	-

POST Request

- Add form ID in request url and specify “/submissions”
- **API key to authorise requests**
- Specify columns by question ID (obtained in Jotform)
“submission[n]”
- Add parameter for each column submission

Request URL

POST ▾ https://eu-api.jotform.com/form/220332334

Headers Body Params (3)

Parameter Assignments

Parameter	Value	Remove
submission[11]	{user_topic}	-
submission[12]	{user_question}	-
submission[13]	{accountEmail}	-

Request URL

POST ▾ https://eu-api.jotform.com/form/220332334801038/submissions

Headers Body Params (3)

Header Assignments

Header	Value	Remove
APIKEY	f126742005e285e3de986aed18b669ef	-

Transform into Variables

Enter object path

APPLY TO Select Variable

POST Request

- Add form ID in request url and specify “/submissions”
- API key to authorise requests
- **Specify columns by question ID (obtained in Jotform)**
“submission[n]”
- Add parameter for each column submission

Request URL

POST ▾ https://eu-api.jotform.com/form/220332334

Headers Body Params (3)

Parameter Assignments

+	submission[11]	-
	VALUE {user_topic}	
+	submission[12]	-
	VALUE {user_question}	
+	submission[13]	-
	VALUE {accountEmail}	

Request URL

POST ▾ https://eu-api.jotform.com/form/220332334

Headers Body Params (3)

Header Assignments

+	APIKEY	-
	VALUE f126742005e285e3de986aed18b669ef	

Transform into Variables

Enter object path

APPLY TO Select Variable

POST Request

- Add form ID in request url and specify “/submissions”
- API key to authorise requests
- Specify columns by question ID (obtained in Jotform)
“submission[n]”
- **Add parameter for each column submission**

Request URL

POST ▾ https://eu-api.jotform.com/form/220332334

Headers Body Params (3)

Parameter Assignments

submission[11]	+	-
VALUE {user_topic}	+	-
submission[12]	+	-
VALUE {user_question}	+	-
submission[13]	+	-
VALUE {accountEmail}	+	-

Request URL

POST ▾ https://eu-api.jotform.com/form/220332334

Headers Body Params (3)

Header Assignments

APIKEY	+	-
VALUE f126742005e285e3de986aed18b669ef	+	-

Transform into Variables

Enter object path

APPLY TO Select Variable

Airtable API

GET Request

- Add base ID and table name in request url
- API key to authorize requests, add “Bearer” before key
“Bearer key123xyz”
- Store response in variable
- Specify columns by name
- Note: returns data 100 records at a time, requires pagination and offsets to read further

Request URL

GET <https://irtable.com/v0/appj3iwsT5t1TjQ5i/credentials>

Headers Params

Header Assignments + -

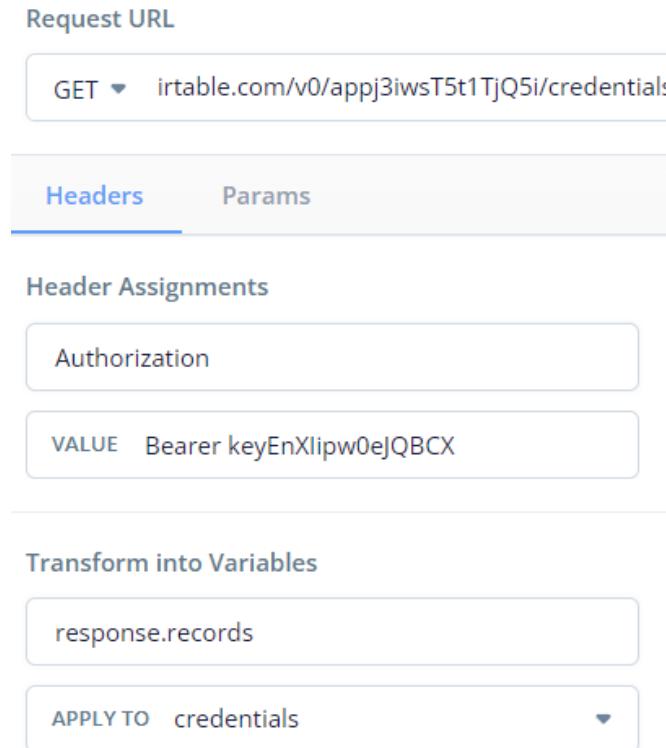
Authorization

VALUE Bearer keyEnXlipw0ejQBCX

Transform into Variables + -

response.records

APPLY TO credentials ▼



```
{
  "records": [
    {
      "id": "rec4PKj4v2RakY7jl",
      "fields": {
        "account_name": "testAccount4",
        "password": "123abc",
        "email": "fraserjiawei@gmail.com",
        "score": "[]"
      },
      "createdTime": "2022-02-10T03:09:10.000Z"
    },
    {
      "id": "rec77x6V2yENCH58N",
      "fields": {
        "account_name": "testAccount10",
        "password": "123abc",
        "email": "fraserjiawei@gmail.com",
        "score": "[]"
      },
      "createdTime": "2022-03-02T20:38:43.000Z"
    },
    {
      "id": "recBnnARbmRNHvNvV",
      "fields": {
        "account_name": "fraser",
        "password": "abc",
        "email": "fraserjiawei@gmail.com",
        "score": "[]"
      },
      "createdTime": "2022-03-07T22:29:02.000Z"
    },
    {
      "id": "recGcdnukayzroZvl",
      "fields": {
        "account_name": "test5",
        "password": "123abc",
        "email": "fraserjiawei@gmail.com",
        "score": "[]"
      },
      "createdTime": "2022-02-10T04:27:31.000Z"
    },
    {
      "id": "recJwJlkH8YC571nh",
      "fields": {
        "account_name": "testAccount3",
        "password": "123",
        "email": "fraserjiawei@gmail.com",
        "score": "[]"
      },
      "createdTime": "2022-01-25T13:35:38.000Z"
    }
  ]
}
```

GET Request

- **Add base ID and table name in request url**
- API key to authorize requests, add “Bearer” before key
“Bearer key123xyz”
- Store response in variable
- Specify columns by name
- Note: returns data 100 records at a time, requires pagination and offsets to read further

Request URL

GET <https://irtable.com/v0/appj3iwsT5t1TjQ5i/credentials>

Headers Params

Header Assignments

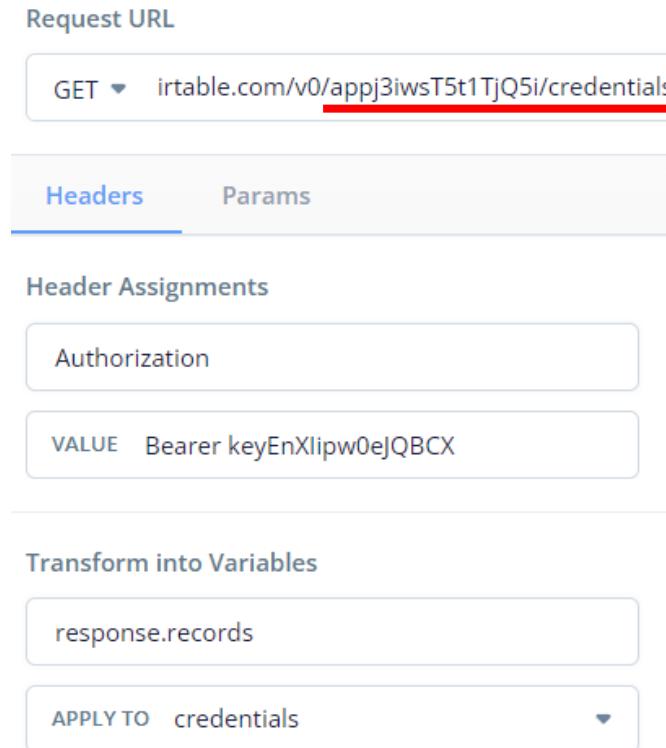
Authorization

VALUE Bearer keyEnXlipw0ejQBCX

Transform into Variables

response.records

APPLY TO credentials



```
{
  "records": [
    {
      "id": "rec4PKj4v2RakY7jl",
      "fields": {
        "account_name": "testAccount4",
        "password": "123abc",
        "email": "fraserjiawei@gmail.com",
        "score": "[]"
      },
      "createdTime": "2022-02-10T03:09:10.000Z"
    },
    {
      "id": "rec77x6V2yENCH58N",
      "fields": {
        "account_name": "testAccount10",
        "password": "123abc",
        "email": "fraserjiawei@gmail.com",
        "score": "[]"
      },
      "createdTime": "2022-03-02T20:38:43.000Z"
    },
    {
      "id": "recBnnARbmRNHvNvV",
      "fields": {
        "account_name": "fraser",
        "password": "abc",
        "email": "fraserjiawei@gmail.com",
        "score": "[]"
      },
      "createdTime": "2022-03-07T22:29:02.000Z"
    },
    {
      "id": "recGcdnukayzroZvl",
      "fields": {
        "account_name": "test5",
        "password": "123abc",
        "email": "fraserjiawei@gmail.com",
        "score": "[]"
      },
      "createdTime": "2022-02-10T04:27:31.000Z"
    },
    {
      "id": "recJwJlkH8YC571nh",
      "fields": {
        "account_name": "testAccount3",
        "password": "123",
        "email": "fraserjiawei@gmail.com",
        "score": "[]"
      },
      "createdTime": "2022-01-25T13:35:38.000Z"
    }
  ]
}
```

GET Request

- Add base ID and table name in request url
- **API key to authorize requests, add “Bearer” before key
“*Bearer key123xyz*”**
- Store response in variable
- Specify columns by name
- Note: returns data 100 records at a time, requires pagination and offsets to read further

Request URL

GET <https://irtable.com/v0/appj3iwsT5t1TjQ5i/credentials>

Headers Params

Header Assignments

Authorization	+	-
VALUE Bearer keyEnXlipw0ejQBCX	+	-

Transform into Variables

response.records	+	-
APPLY TO credentials	▼	

```
{
  "records": [
    {
      "id": "rec4PKj4v2RakY7jl",
      "fields": {
        "account_name": "testAccount4",
        "password": "123abc",
        "email": "fraserjiawei@gmail.com",
        "score": "[]"
      },
      "createdTime": "2022-02-10T03:09:10.000Z"
    },
    {
      "id": "rec77x6V2yENCH58N",
      "fields": {
        "account_name": "testAccount10",
        "password": "123abc",
        "email": "fraserjiawei@gmail.com",
        "score": "[]"
      },
      "createdTime": "2022-03-02T20:38:43.000Z"
    },
    {
      "id": "recBnnARbmRNHvNvV",
      "fields": {
        "account_name": "fraser",
        "password": "abc",
        "email": "fraserjiawei@gmail.com",
        "score": "[]"
      },
      "createdTime": "2022-03-07T22:29:02.000Z"
    },
    {
      "id": "recGdnukayzroZvl",
      "fields": {
        "account_name": "test5",
        "password": "123abc",
        "email": "fraserjiawei@gmail.com",
        "score": "[]"
      },
      "createdTime": "2022-02-10T04:27:31.000Z"
    },
    {
      "id": "recJwJlkH8YC571nh",
      "fields": {
        "account_name": "testAccount3",
        "password": "123",
        "email": "fraserjiawei@gmail.com",
        "score": "[]"
      },
      "createdTime": "2022-01-25T13:35:38.000Z"
    }
  ]
}
```

GET Request

- Add base ID and table name in request url
- API key to authorize requests, add “Bearer” before key
“*Bearer key123xyz*”
- **Store response in variable**
- Specify columns by name
- Note: returns data 100 records at a time, requires pagination and offsets to read further

Request URL

GET <https://irtable.com/v0/appj3iwsT5t1TjQ5i/credentials>

Headers Params

Header Assignments

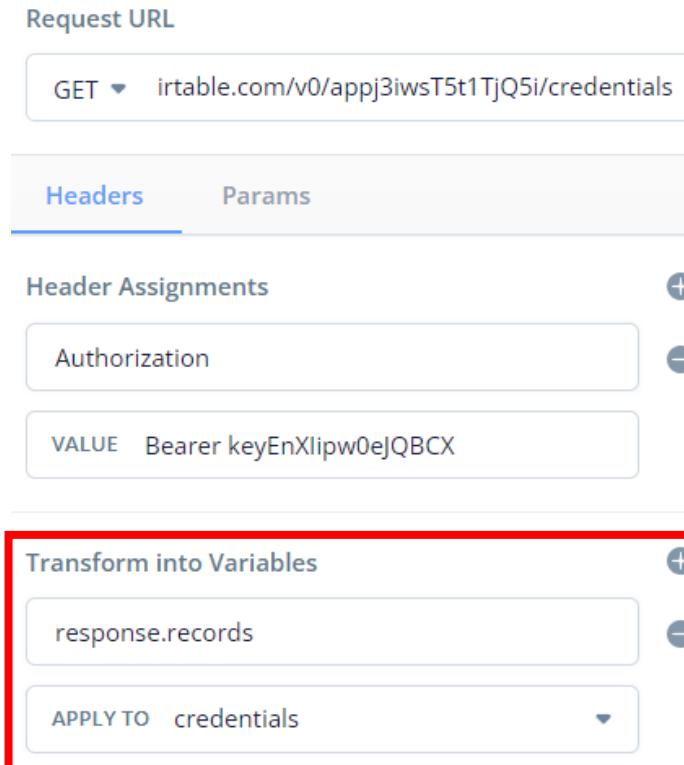
Authorization

VALUE Bearer keyEnXlipw0ejQBCX

Transform into Variables

response.records

APPLY TO credentials



```
{
  "records": [
    {
      "id": "rec4PKj4v2RakY7jl",
      "fields": {
        "account_name": "testAccount4",
        "password": "123abc",
        "email": "fraserjiawei@gmail.com",
        "score": "[]"
      },
      "createdTime": "2022-02-10T03:09:10.000Z"
    },
    {
      "id": "rec77x6V2yENCH58N",
      "fields": {
        "account_name": "testAccount10",
        "password": "123abc",
        "email": "fraserjiawei@gmail.com",
        "score": "[]"
      },
      "createdTime": "2022-03-02T20:38:43.000Z"
    },
    {
      "id": "recBnnARbmRNHvNvV",
      "fields": {
        "account_name": "fraser",
        "password": "abc",
        "email": "fraserjiawei@gmail.com",
        "score": "[]"
      },
      "createdTime": "2022-03-07T22:29:02.000Z"
    },
    {
      "id": "recGdnukayzroZvl",
      "fields": {
        "account_name": "test5",
        "password": "123abc",
        "email": "fraserjiawei@gmail.com",
        "score": "[]"
      },
      "createdTime": "2022-02-10T04:27:31.000Z"
    },
    {
      "id": "recJwJlkH8YC571nh",
      "fields": {
        "account_name": "testAccount3",
        "password": "123",
        "email": "fraserjiawei@gmail.com",
        "score": "[]"
      },
      "createdTime": "2022-01-25T13:35:38.000Z"
    }
  ]
}
```

GET Request

- Add base ID and table name in request url
- API key to authorize requests, add “Bearer” before key
“*Bearer key123xyz*”
- Store response in variable
- **Specify columns by name**
- Note: returns data 100 records at a time, requires pagination and offsets to read further

Request URL

GET

Headers

Params

Header Assignments

Authorization

VALUE

Transform into Variables

response.records

APPLY TO

```
{  
  "records": [  
    {  
      "id": "rec4PKj4v2RakY7j1",  
      "fields": {  
        "account_name": "testAccount4",  
        "password": "123abc",  
        "email": "fraserjiawei@gmail.com",  
        "score": "[]"  
      },  
      "createdTime": "2022-02-10T03:09:10.000Z"  
    },  
    {  
      "id": "rec77x6V2yENCH58N",  
      "fields": {  
        "account_name": "testAccount10",  
        "password": "123abc",  
        "email": "fraserjiawei@gmail.com",  
        "score": "[]"  
      },  
      "createdTime": "2022-03-02T20:38:43.000Z"  
    },  
    {  
      "id": "recBnnARbmRMHvNvV",  
      "fields": {  
        "account_name": "fraser",  
        "password": "abc",  
        "email": "fraserjiawei@gmail.com",  
        "score": "[]"  
      },  
      "createdTime": "2022-03-07T22:29:02.000Z"  
    },  
    {  
      "id": "recGcdnukayzroZvl",  
      "fields": {  
        "account_name": "test5",  
        "password": "123abc",  
        "email": "fraserjiawei@gmail.com",  
        "score": "[]"  
      },  
      "createdTime": "2022-02-10T04:27:31.000Z"  
    },  
    {  
      "id": "recJwJIKh8YC571nh",  
      "fields": {  
        "account_name": "testAccount3",  
        "password": "123",  
        "email": "fraserjiawei@gmail.com",  
        "score": "[]"  
      },  
      "createdTime": "2022-01-25T13:35:38.000Z"  
    },  
  ]  
}
```

Array of records

Individual record

Columns and stored values

GET Request

- Add base ID and table name in request url
- API key to authorize requests, add “Bearer” before key
“Bearer key123xyz”
- Store response in variable
- Specify columns by name
- **Note: returns data 100 records at a time, requires pagination and offsets to read further**

Request URL

GET <https://irtable.com/v0/appj3iwsT5t1TjQ5i/credentials>

Headers Params

Header Assignments

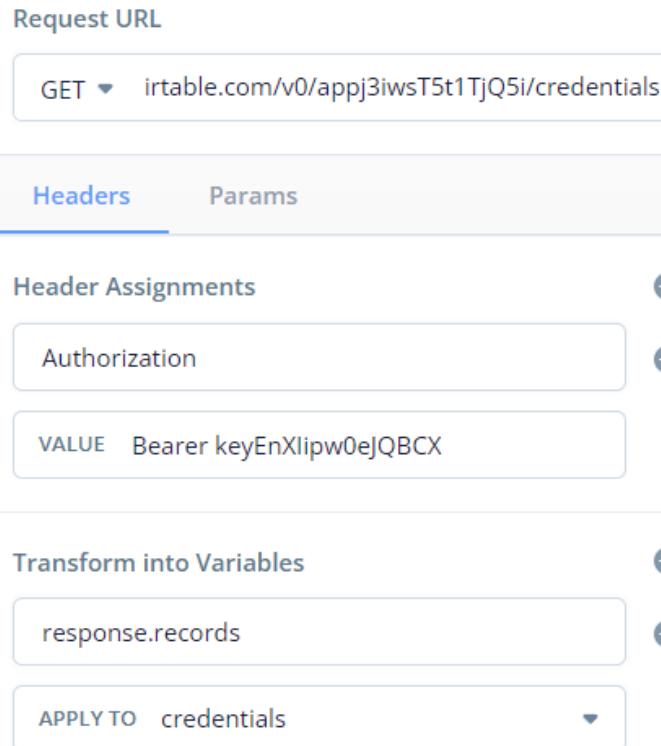
Authorization

VALUE Bearer keyEnXlipw0ejQBCX

Transform into Variables

response.records

APPLY TO credentials



```
{  
  "records": [  
    {  
      "id": "rec4PKj4v2RakY7jl",  
      "fields": {  
        "account_name": "testAccount4",  
        "password": "123abc",  
        "email": "fraserjiawei@gmail.com",  
        "score": "[]"  
      },  
      "createdTime": "2022-02-10T03:09:10.000Z"  
    },  
    {  
      "id": "rec77x6V2yENCH58N",  
      "fields": {  
        "account_name": "testAccount10",  
        "password": "123abc",  
        "email": "fraserjiawei@gmail.com",  
        "score": "[]"  
      },  
      "createdTime": "2022-03-02T20:38:43.000Z"  
    },  
    {  
      "id": "recBnnARbmRNHvNvV",  
      "fields": {  
        "account_name": "fraser",  
        "password": "abc",  
        "email": "fraserjiawei@gmail.com",  
        "score": "[]"  
      },  
      "createdTime": "2022-03-07T22:29:02.000Z"  
    },  
    {  
      "id": "recGdnukayzroZvl",  
      "fields": {  
        "account_name": "test5",  
        "password": "123abc",  
        "email": "fraserjiawei@gmail.com",  
        "score": "[]"  
      },  
      "createdTime": "2022-02-10T04:27:31.000Z"  
    },  
    {  
      "id": "recJwJlkH8YCs71nh",  
      "fields": {  
        "account_name": "testAccount3",  
        "password": "123",  
        "email": "fraserjiawei@gmail.com",  
        "score": "[]"  
      },  
      "createdTime": "2022-01-25T13:35:38.000Z"  
    },  
    ...  
  ]  
}
```

1

2

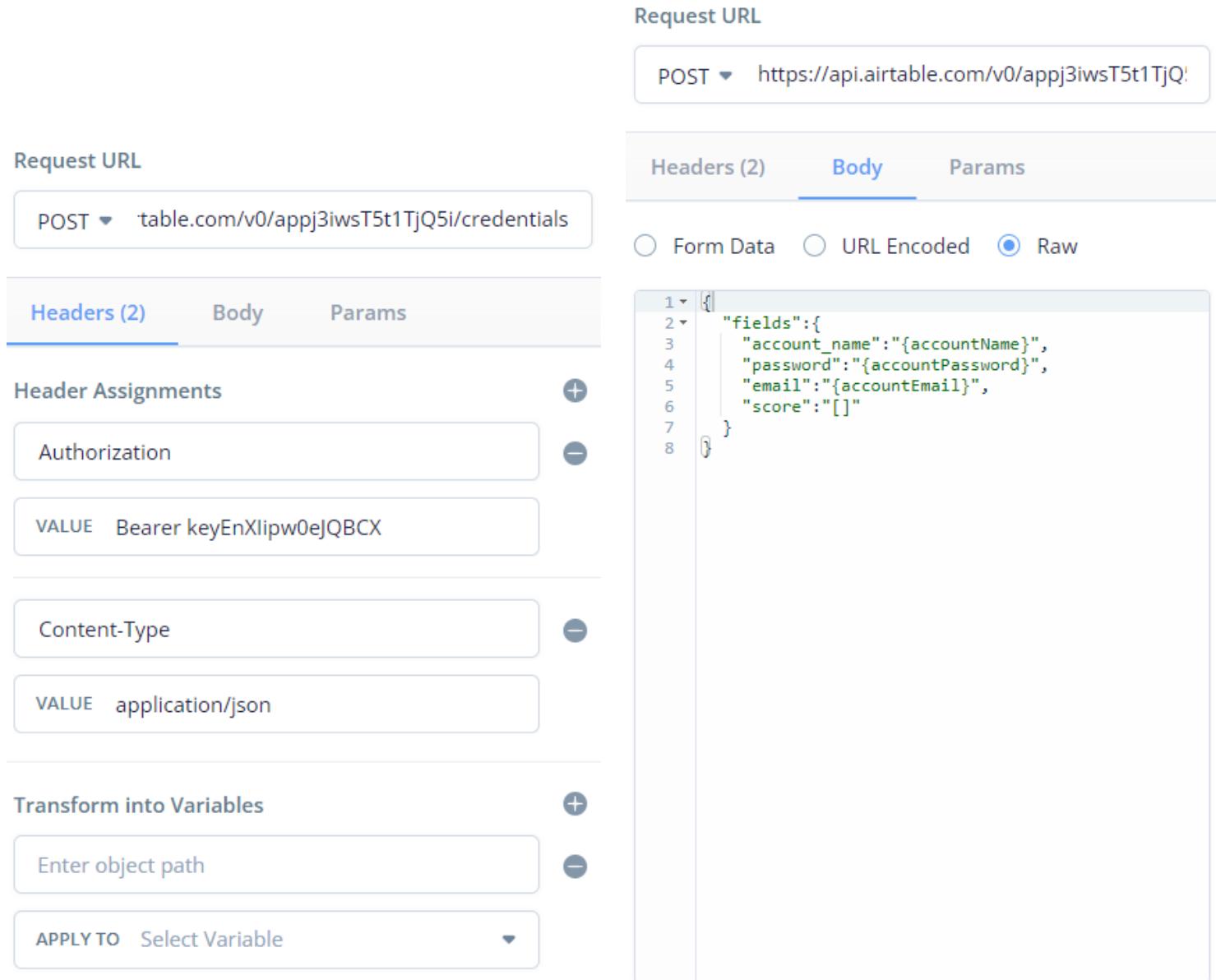
3

4

5

POST Request

- Add table ID and table name in request url
- API key to authorize requests, add “Bearer” before key *“Bearer key123xyz”*
- Add “Content-Type” header with value “application/json”
- Add content in url body with raw formatting
- JSON object “fields” containing each column name and its value



Request URL

POST <https://api.airtable.com/v0/appj3iwsT5t1TjQ5i/credentials>

Headers (2) Body Params

Form Data URL Encoded Raw

```
1  {
2   "fields":{
3     "account_name": "{accountName}",
4     "password": "{accountPassword}",
5     "email": "{accountEmail}",
6     "score": "[]"
7   }
8 }
```

POST Request

- **Add table ID and table name in request url**
- API key to authorize requests, add “Bearer” before key *“Bearer key123xyz”*
- Add “Content-Type” header with value “application/json”
- Add content in url body with raw formatting
- JSON object “fields” containing each column name and its value

Request URL

POST ▾ <https://api.airtable.com/v0/appj3iwsT5t1TjQ5i/credentials>

Headers (2) Body Params

Form Data URL Encoded Raw

Header Assignments

Authorization

VALUE Bearer keyEnXlipw0eJQBCX

Content-Type

VALUE application/json

Transform into Variables

Enter object path

APPLY TO Select Variable

```
1 {  
2   "fields": {  
3     "account_name": "{accountName}",  
4     "password": "{accountPassword}",  
5     "email": "{accountEmail}",  
6     "score": "[]"  
7   }  
8 }
```

POST Request

- Add table ID and table name in request url
- **API key to authorize requests, add “Bearer” before key *“Bearer key123xyz”***
- Add “Content-Type” header with value “application/json”
- Add content in url body with raw formatting
- JSON object “fields” containing each column name and its value

Request URL

POST ▾ https://api.airtable.com/v0/appj3iwsT5t1TjQ!

Request URL

POST ▾ table.com/v0/appj3iwsT5t1TjQ5i/credentials

Headers (2) Body Params

Form Data URL Encoded Raw

Header Assignments

Authorization

VALUE Bearer keyEnXlipw0eJQBCX

Content-Type

VALUE application/json

Transform into Variables

Enter object path

APPLY TO Select Variable

1 "fields":{
2 "account_name":"{accountName}",
3 "password":"{accountPassword}",
4 "email":"{accountEmail}",
5 "score":[]
6 }
7
8

POST Request

- Add table ID and table name in request url
- API key to authorize requests, add “Bearer” before key *“Bearer key123xyz”*
- **Add “Content-Type” header with value “application/json”**
- Add content in url body with raw formatting
- JSON object “fields” containing each column name and its value

Request URL

POST ▾ https://api.airtable.com/v0/appj3iwsT5t1TjQ!

Request URL

POST ▾ table.com/v0/appj3iwsT5t1TjQ5i/credentials

Headers (2) Body Params

Form Data URL Encoded Raw

Header Assignments

Authorization

VALUE Bearer keyEnXlipw0eJQBCX

Content-Type

VALUE application/json

Transform into Variables

Enter object path

APPLY TO Select Variable

1 "fields":{
2 "account_name":"{accountName}",
3 "password":"{accountPassword}",
4 "email":"{accountEmail}",
5 "score":[]
6 }
7
8

POST Request

- Add table ID and table name in request url
- API key to authorize requests, add “Bearer” before key *“Bearer key123xyz”*
- Add “Content-Type” header with value “application/json”
- **Add content in url body with raw formatting**
- JSON object “fields” containing each column name and its value

Request URL: POST <https://api.airtable.com/v0/appj3iwsT5t1TjQ5i/credentials>

Headers (2):

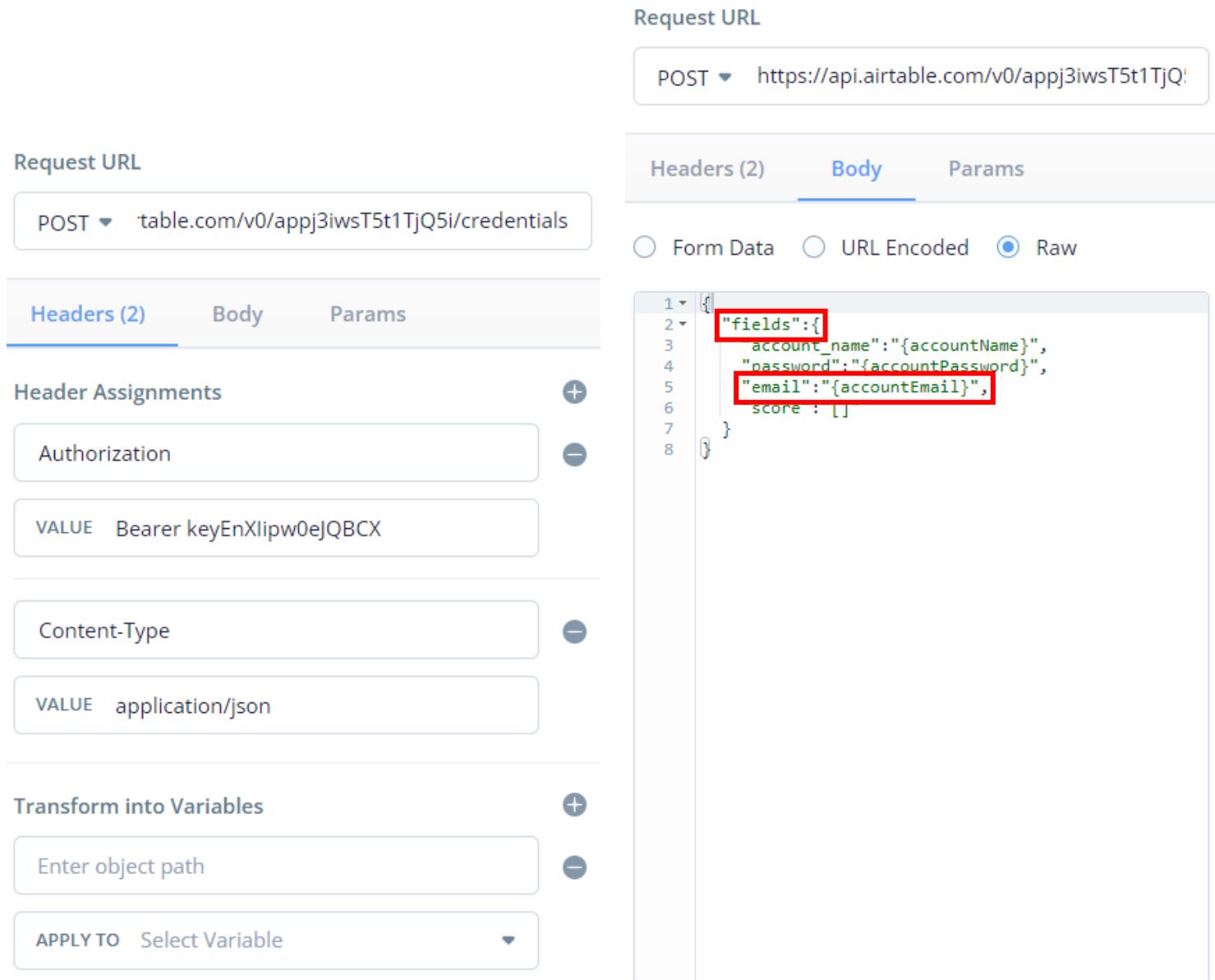
- Authorization: Bearer keyEnXlipw0eJQBCX
- Content-Type: application/json

Body (Raw):

```
{ "fields": { "account_name": "{accountName}", "password": "{accountPassword}", "email": "{accountEmail}", "score": "[]" } }
```

POST Request

- Add table ID and table name in request url
- API key to authorize requests, add “Bearer” before key *“Bearer key123xyz”*
- Add “Content-Type” header with value “application/json”
- Add content in url body with raw formatting
- **JSON object “fields” containing each column name and its value**



Request URL: POST <https://api.airtable.com/v0/appj3iwsT5t1TjQ5i/credentials>

Headers (2):

- Authorization: Bearer keyEnXlipw0ejQBCX
- Content-Type: application/json

Body (Raw):

```
1 {  
2   "fields": {  
3     "account_name": "{accountName}",  
4     "password": "{accountPassword}",  
5     "email": "{accountEmail}",  
6     "score": []  
7   }  
8 }
```

PATCH Request

- Add table ID and table name in request url
- API key to authorize requests, add “Bearer” before key *“Bearer key123xyz”*
- Add “Content-Type” header with value “application/json”
- Add content in url body with raw formatting
- JSON array “records” containing the record and the information inside it to update
- Refer to each record by its “id” property (found on Airtable and in GET requests)

Request URL

PATCH ▾ https://api.airtable.com/v0/appj3iwsT5t1TjQ5i/credentials

Headers (2) Body Params

Form Data URL Encoded Raw

Header Assignments

Authorization

VALUE Bearer keyEnXlipw0eJQBCX

Content-Type

VALUE application/json

Transform into Variables

Enter object path

APPLY TO Select Variable

```
1 [ { "records": [ 2 { 3 { 4 { "id": { "accountID": 5 "fields": 6 { 7 { "score": { "user_data": 8 } 9 } } } } } ] }
```

PATCH Request

- **Add table ID and table name in request url**
- API key to authorize requests, add “Bearer” before key **“Bearer key123xyz”**
- Add “Content-Type” header with value “application/json”
- Add content in url body with raw formatting
- JSON array “records” containing the record and the information inside it to update
- Refer to each record by its “id” property (found on Airtable and in GET requests)

Request URL

PATCH ▾ https://api.airtable.com/v0/appj3iwsT5t1TjQ5i/credentials

Headers (2) Body Params

Form Data URL Encoded Raw

```
1  {
2    "records": [
3      {
4        "id": {accountID},
5        "fields": [
6          {
7            "score": {user_data}
8          }
9        ]
5      }
6    ]
7  }
```

PATCH Request

- Add table ID and table name in request url
- **API key to authorize requests, add “Bearer” before key *“Bearer key123xyz”***
- Add “Content-Type” header with value “application/json”
- Add content in url body with raw formatting
- JSON array “records” containing the record and the information inside it to update
- Refer to each record by its “id” property (found on Airtable and in GET requests)

Request URL

PATCH ▾ https://api.airtable.com/v0/appj3iwsT5t1TjQ5i/credentials

Headers (2) Body Params

Form Data URL Encoded Raw

Header Assignments

Authorization	+
VALUE Bearer keyEnXlipw0eJQBCX	-

Content-Type

application/json	-
------------------	---

Transform into Variables

Enter object path	+
APPLY TO Select Variable	-

Request URL

PATCH ▾ https://api.airtable.com/v0/appj3iwsT5t1TjQ5i/credentials

Headers (2) Body Params

Form Data URL Encoded Raw

```
1 [
2   "records": [
3     {
4       "id": {accountID},
5       "fields": [
6         {
7           "score": {user_data}
8         }
9       ]
7     }
  ]
]
```

PATCH Request

- Add table ID and table name in request url
- API key to authorize requests, add “Bearer” before key *“Bearer key123xyz”*
- **Add “Content-Type” header with value “application/json”**
- Add content in url body with raw formatting
- JSON array “records” containing the record and the information inside it to update
- Refer to each record by its “id” property (found on Airtable and in GET requests)

Request URL

PATCH ▾ https://api.airtable.com/v0/appj3iwsT5t1TjQ5i/credentials

Headers (2) Body Params

Form Data URL Encoded Raw

Header Assignments

Authorization

VALUE Bearer keyEnXlipw0eJQBCX

Content-Type

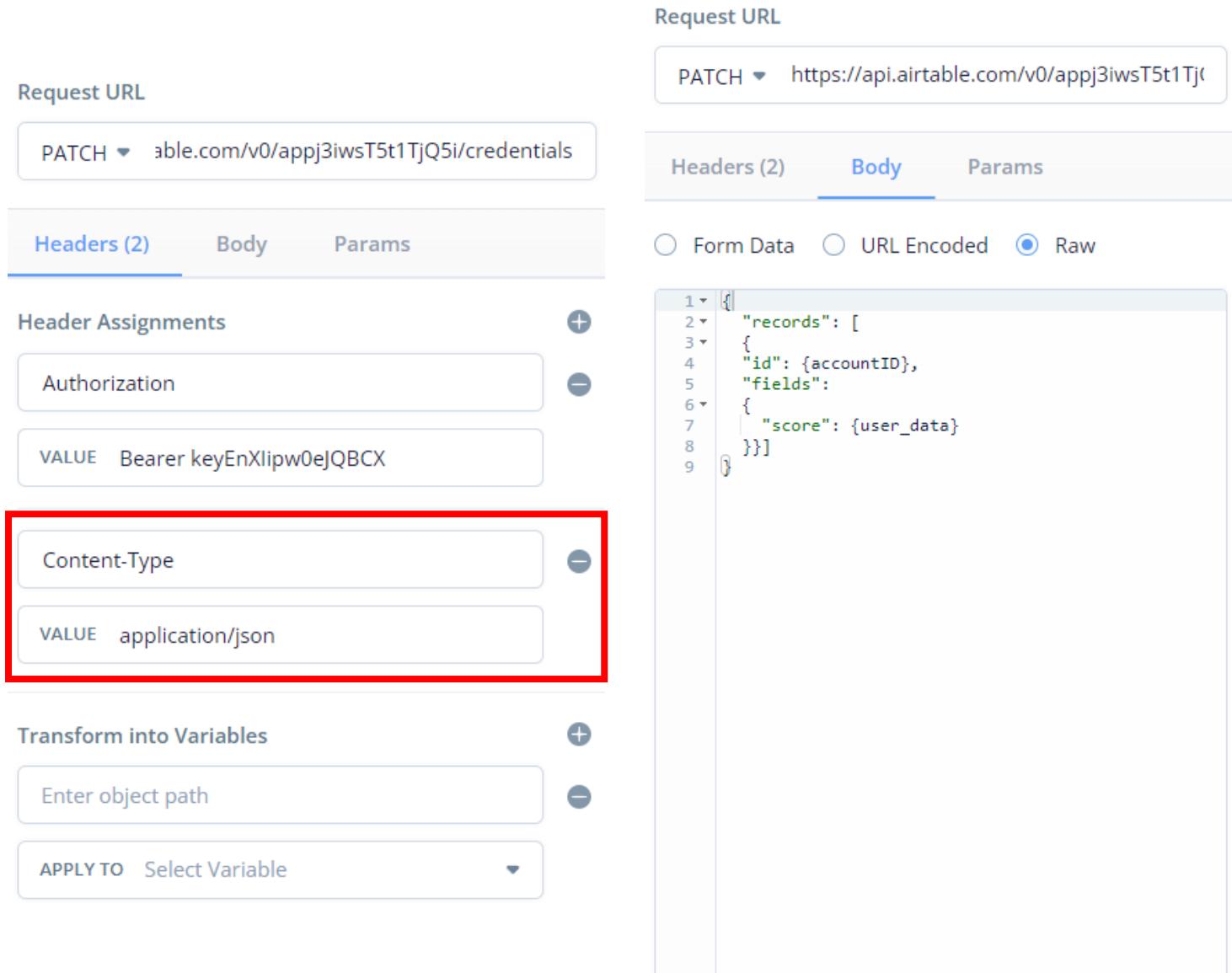
VALUE application/json

Transform into Variables

Enter object path

APPLY TO Select Variable

1 [2 "records": [3 { 4 "id": {accountID}, 5 "fields": 6 { 7 "score": {user_data} 8 }}] 9]



PATCH Request

- Add table ID and table name in request url
- API key to authorize requests, add “Bearer” before key **“Bearer key123xyz”**
- Add “Content-Type” header with value “application/json”
- **Add content in url body with raw formatting**
- JSON array “records” containing the record and the information inside it to update
- Refer to each record by its “id” property (found on Airtable and in GET requests)

Request URL

PATCH ▾ https://api.airtable.com/v0/appj3iwsT5t1TjQ5i/credentials

Headers (2) **Body** Params

Form Data URL Encoded Raw

Header Assignments

Authorization

VALUE Bearer keyEnXlipw0eJQBCX

Content-Type

VALUE application/json

Transform into Variables

Enter object path

APPLY TO Select Variable

Request URL

PATCH ▾ https://api.airtable.com/v0/appj3iwsT5t1TjQ5i/credentials

Headers (2) **Body** Params

Form Data URL Encoded Raw

```
1  {
2    "records": [
3      {
4        "id": {accountID},
5        "fields": [
6          {
7            "score": {user_data}
8          }
9        ]
5      }
6    ]
7  }
```

PATCH Request

- Add table ID and table name in request url
- API key to authorize requests, add “Bearer” before key *“Bearer key123xyz”*
- Add “Content-Type” header with value “application/json”
- Add content in url body with raw formatting
- **JSON array “records” containing the record and the information inside it to update**
- Refer to each record by its “id” property (found on Airtable and in GET requests)

Request URL

PATCH ▾ https://api.airtable.com/v0/appj3iwsT5t1TjQ5i/credentials

Headers (2) Body Params

Form Data URL Encoded Raw

Header Assignments

Authorization

VALUE Bearer keyEnXlipw0eJQBCX

Content-Type

VALUE application/json

Transform into Variables

Enter object path

APPLY TO Select Variable

Request URL

PATCH ▾ https://api.airtable.com/v0/appj3iwsT5t1TjQ5i/credentials

Headers (2) Body Params

Form Data URL Encoded Raw

```
1 [
2   "records": [
3     {
4       "id": {accountID},
5       "fields": [
6         {
7           "score": {user_data}
8         }
9       ]
7     }
  ]
]
```

PATCH Request

- Add table ID and table name in request url
- API key to authorize requests, add “Bearer” before key **“Bearer key123xyz”**
- Add “Content-Type” header with value “application/json”
- Add content in url body with raw formatting
- JSON array “records” containing the record and the information inside it to update
- **Refer to each record by its “id” property (found on Airtable and in GET requests)**

Request URL

PATCH ▾ https://api.airtable.com/v0/appj3iwsT5t1TjQ5i/credentials

Headers (2) Body Params

Form Data URL Encoded Raw

Header Assignments

Authorization

VALUE Bearer keyEnXlipw0eJQBCX

Content-Type

VALUE application/json

Transform into Variables

Enter object path

APPLY TO Select Variable

1 {
2 "records": [
3 {
4 "id": {accountID},
5 "fields":
6 {
7 "score": {user_data}
8 }}]
9 }

PUT Request

- Add table ID and table name in request url
- API key to authorize requests, add “Bearer” before key **“Bearer key123xyz”**
- Add “Content-Type” header with value “application/json”
- Add content in url body with raw formatting
- JSON array “records” containing the record to replace the old one with
- Refer to each record by its “id” property (found on Airtable and in GET requests)

Request URL

PUT <https://api.airtable.com/v0/appj3iwsT5t1TjQ5i>

Headers (2) **Body** Params

Form Data URL Encoded Raw

Header Assignments

Authorization

VALUE Bearer keyEnXlipw0ejQBCX

Content-Type

VALUE application/json

Transform into Variables

Enter object path

APPLY TO Select Variable

```
1 [
2   "records": [
3     {
4       "id": "recNmzzZqLwEZnQ0X",
5       "fields": [
6         {
7           "Scores": {average_data},
8           "Number": {average_num}
9         }
10      ]
11    }
]
```

PUT Request

- **Add table ID and table name in request url**
- API key to authorize requests, add “Bearer” before key **“Bearer key123xyz”**
- Add “Content-Type” header with value “application/json”
- Add content in url body with raw formatting
- JSON array “records” containing the record to replace the old one with
- Refer to each record by its “id” property (found on Airtable and in GET requests)

Request URL

PUT <https://api.airtable.com/v0/appj3iwsT5t1TjQ5i>

Headers (2) **Body** Params

Form Data URL Encoded Raw

Header Assignments

Authorization

VALUE Bearer keyEnXlipw0ejQBCX

Content-Type

VALUE application/json

Transform into Variables

Enter object path

APPLY TO Select Variable

```
1 [
2   "records": [
3     {
4       "id": "recNmzzZqLwEZNQOX",
5       "fields": [
6         {
7           "Scores": {average_data},
8           "Number": {average_num}
9         }
10      ]
11    }
]
```

PUT Request

- Add table ID and table name in request url
- **API key to authorize requests, add “Bearer” before key “*Bearer key123xyz*”**
- Add “Content-Type” header with value “application/json”
- Add content in url body with raw formatting
- JSON array “records” containing the record to replace the old one with
- Refer to each record by its “id” property (found on Airtable and in GET requests)

Request URL

PUT ▾ <https://api.airtable.com/v0/appj3iwsT5t1TjQ5i>

Headers (2) **Body** Params

Form Data URL Encoded Raw

Header Assignments

Authorization	+
VALUE Bearer keyEnXlipw0ejQBCX	-

Content-Type

application/json	-
------------------	---

Transform into Variables

Enter object path	+
APPLY TO Select Variable	-

1 [
2 "records": [
3 {
4 "id": "recNmzzZqLwEZNQOX",
5 "fields":
6 {
7 "Scores": {average_data},
8 "Number": {average_num}
9 }
10 }]
11]

PUT Request

- Add table ID and table name in request url
- API key to authorize requests, add “Bearer” before key “*Bearer key123xyz*”
- **Add “Content-Type” header with value “application/json”**
- Add content in url body with raw formatting
- JSON array “records” containing the record to replace the old one with
- Refer to each record by its “id” property (found on Airtable and in GET requests)

Request URL

PUT <https://api.airtable.com/v0/appj3iwsT5t1TjQ5i>

Headers (2) **Body** Params

Form Data URL Encoded Raw

Header Assignments

Authorization

VALUE Bearer keyEnXlipw0ejQBCX

Content-Type

VALUE application/json

Transform into Variables

Enter object path

APPLY TO Select Variable

```
1 [
2   "records": [
3     {
4       "id": "recNmZZqLwEZnQOX",
5       "fields": [
6         {
7           "Scores": {average_data},
8           "Number": {average_num}
9         }
10      ]
11    }
]
```

PUT Request

- Add table ID and table name in request url
- API key to authorize requests, add “Bearer” before key **“Bearer key123xyz”**
- Add “Content-Type” header with value “application/json”
- **Add content in url body with raw formatting**
- JSON array “records” containing the record to replace the old one with
- Refer to each record by its “id” property (found on Airtable and in GET requests)

Request URL

PUT <https://api.airtable.com/v0/appj3iwsT5t1TjQ5i>

Headers (2) **Body** Params

Form Data URL Encoded Raw

Header Assignments

Authorization

VALUE Bearer keyEnXlipw0ejQBCX

Content-Type

VALUE application/json

Transform into Variables

Enter object path

APPLY TO Select Variable

```
1 [
2   "records": [
3     {
4       "id": "recNmzzZqLwEZnQ0X",
5       "fields": [
6         {
7           "Scores": {average_data},
8           "Number": {average_num}
9         }
10      ]
11    }
12  ]
13]
```

PUT Request

- Add table ID and table name in request url
- API key to authorize requests, add “Bearer” before key **“Bearer key123xyz”**
- Add “Content-Type” header with value “application/json”
- Add content in url body with raw formatting
- **JSON array “records” containing the record to replace the old one with**
- Refer to each record by its “id” property (found on Airtable and in GET requests)

Request URL

PUT <https://api.airtable.com/v0/appj3iwsT5t1TjQ5i>

Headers (2) **Body** Params

Form Data URL Encoded Raw

Header Assignments

Authorization

VALUE Bearer keyEnXlipw0ejQBCX

Content-Type

VALUE application/json

Transform into Variables

Enter object path

APPLY TO Select Variable

Request URL

PUT <https://api.airtable.com/v0/appj3iwsT5t1TjQ5i/averages>

Headers (2) **Body** Params

Form Data URL Encoded Raw

1
2
3
4
5
6
7
8
9
10
11

```
1 "records": [  
2   {"id": "recNmZZqLwEZNQOX",  
3    "fields":  
4      {"Scores": {"average_data"},  
5       "Number": {"average_num"}  
6     }  
7   ]  
8 ]  
9 }  
10 }  
11 }
```

Replacement record

PUT Request

- Add table ID and table name in request url
- API key to authorize requests, add “Bearer” before key **“Bearer key123xyz”**
- Add “Content-Type” header with value “application/json”
- Add content in url body with raw formatting
- JSON array “records” containing the record to replace the old one with
- **Refer to each record by its “id” property (found on Airtable and in GET requests)**

Request URL

PUT <https://api.airtable.com/v0/appj3iwsT5t1TjQ5i>

Headers (2) **Body** Params

Form Data URL Encoded Raw

Header Assignments

Authorization

VALUE Bearer keyEnXlipw0ejQBCX

Content-Type

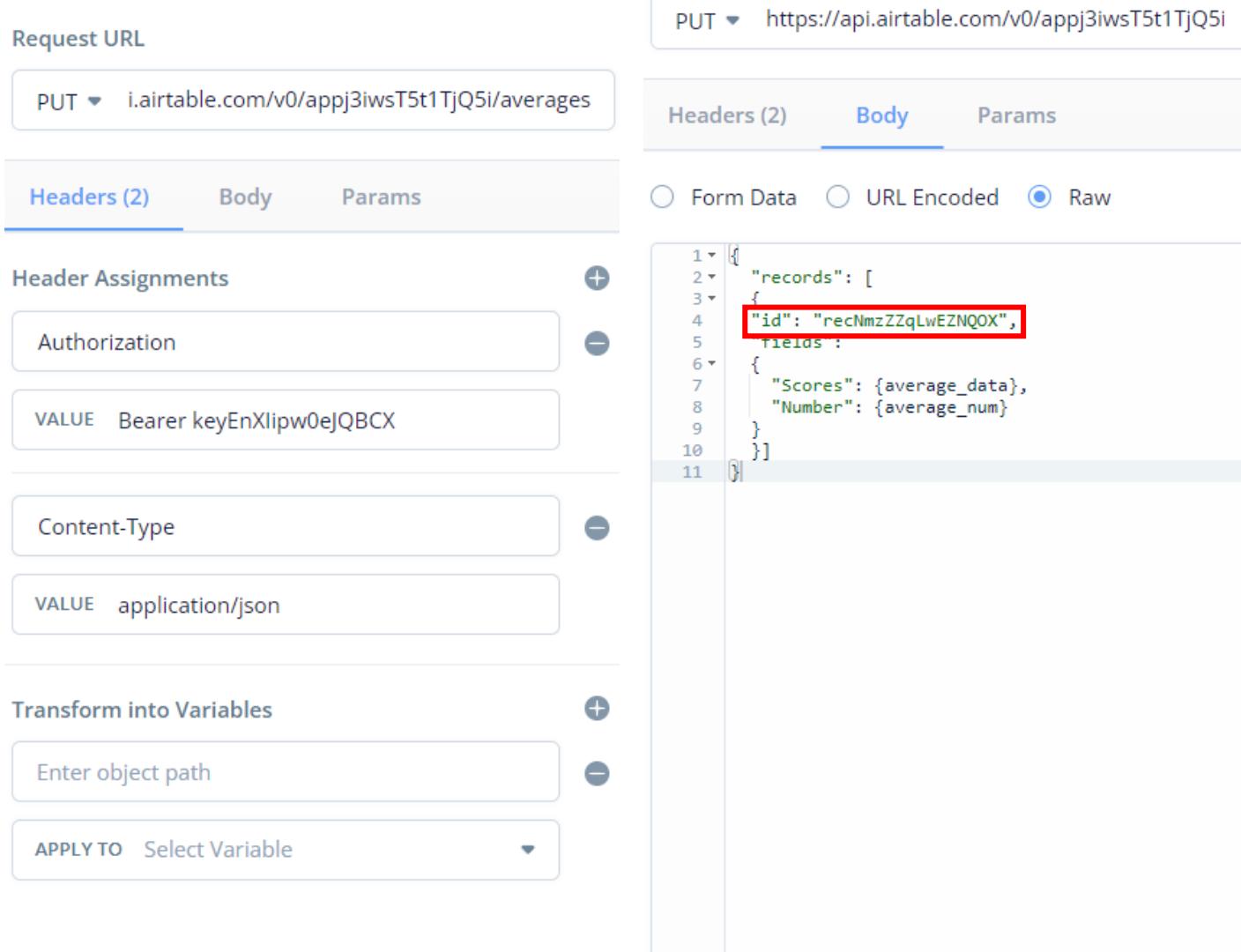
VALUE application/json

Transform into Variables

Enter object path

APPLY TO Select Variable

```
1 [
2   "records": [
3     {
4       "id": "recNmZZqLwEZnQ0X",
5       "fields": [
6         {
7           "Scores": {average_data},
8           "Number": {average_num}
9         }
10      ]
11    }
12  ]
13]
```



QuickChart API

POST Request

- Add “Content-Type” header with value “application/json”
- Specify the graph in the url body with raw formatting
- Save “response.url” to variable
- Upon clicking the url, the graph begins to be rendered
- Note: may not show instantly for complicated graphs

Request URL

POST <https://quickchart.io/chart/create>

Headers Body Params

Header Assignments

Content-Type

VALUE application/json

Transform into Variables

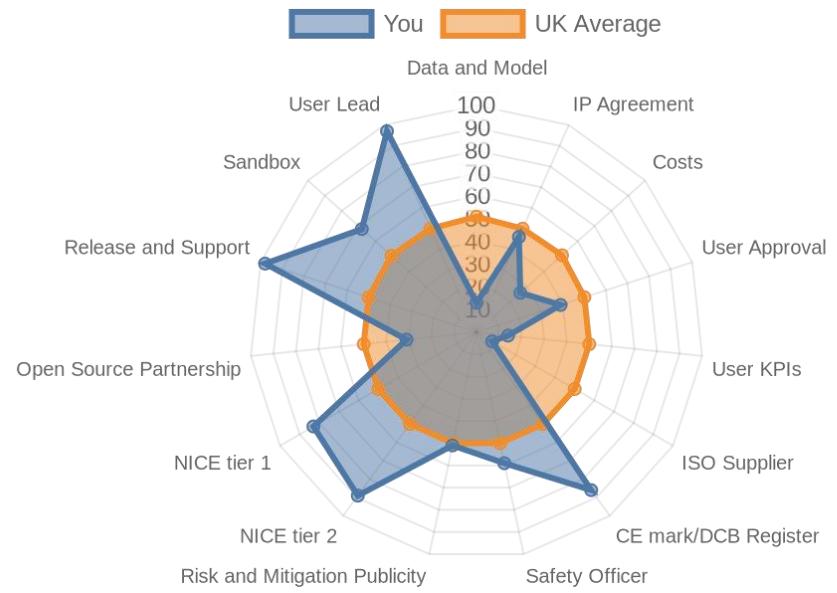
response.url

APPLY TO url

Headers Body Params

Form Data URL Encoded Raw

```
1  {
2    "chart": {
3      "type": "radar",
4      "data": {
5        "labels": {graph_labels},
6        "datasets": [
7          {
8            "label": "You",
9            "data": {user_data}
10         },
11         {
12           "label": "UK Average",
13           "data": {average_data}
14         }
15       ],
16       "options": {
17         "scale": {
18           "ticks": {
19             "min": 0,
20             "max": 100,
21             "stepSize": 10
22           }
23         }
24       }
25     }
26   }
```



POST Request

- Add “Content-Type” header with value “application/json”
- Specify the graph in the url body with raw formatting
- Save “response.url” to variable
- Upon clicking the url, the graph begins to be rendered
- Note: may not show instantly for complicated graphs

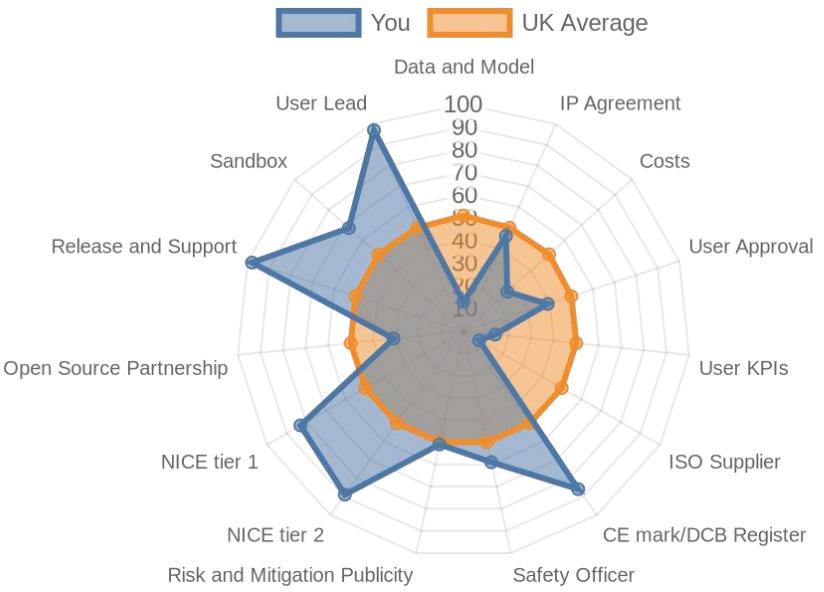
```
Request URL
POST https://quickchart.io/chart/create

Headers Body Params

Header Assignments
Content-Type
VALUE application/json

Transform into Variables
response.url
APPLY TO url

Headers Body Params
Form Data URL Encoded Raw
1  [
2    "chart": {
3      "type": "radar",
4      "data": {
5        "labels": {graph_labels},
6        "datasets": [
7          {
8            "label": "You",
9            "data": {user_data}
10           },
11          {
12            "label": "UK Average",
13            "data": {average_data}
14          }
15        ],
16        "options" : {
17          "scale" : {
18            "ticks" : {
19              "min" : 0,
20              "max" : 100,
21              "stepSize" : 10
22            }
23          }
24        }
25      }
26    ]
27  ]
```



POST Request

- Add “Content-Type” header with value “application/json”
- **Specify the graph in the url body with raw formatting**
- Save “response.url” to variable
- Upon clicking the url, the graph begins to be rendered
- Note: may not show instantly for complicated graphs

Request URL

POST ▾ https://quickchart.io/chart/create

Headers Body Params

Header Assignments

Content-Type

VALUE application/json

Transform into Variables

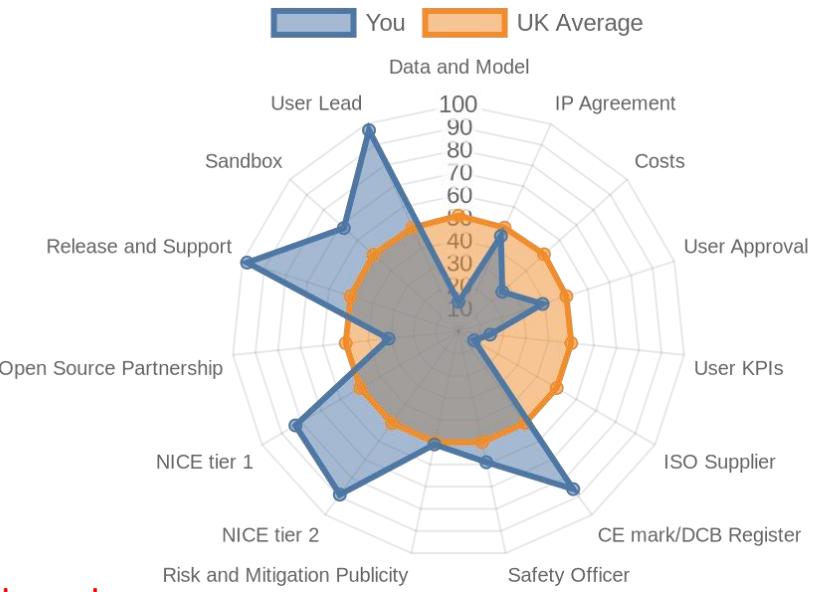
response.url

APPLY TO url

Headers Body Params

Form Data URL Encoded Raw

```
1  {
2    "chart": {
3      "type": "radar", Graph type
4      "data": [
5        {
6          "label": "You", Vertices of the radar graph
7          "data": {user_data}
8        },
9        {
10          "label": "UK Average", Name and array of
11          "data": {average_data} data for each vertex
12        }
13      ],
14      "options": {
15        "scale": {
16          "ticks": {
17            "min": 0, Additional scale
18            "max": 100, adjustments
19            "stepSize": 10
20          }
21        }
22      }
23    }
24  }
```



POST Request

- Add “Content-Type” header with value “application/json”
- Specify the graph in the url body with raw formatting
- **Save “response.url” to variable**
- Upon clicking the url, the graph begins to be rendered
- Note: may not show instantly for complicated graphs

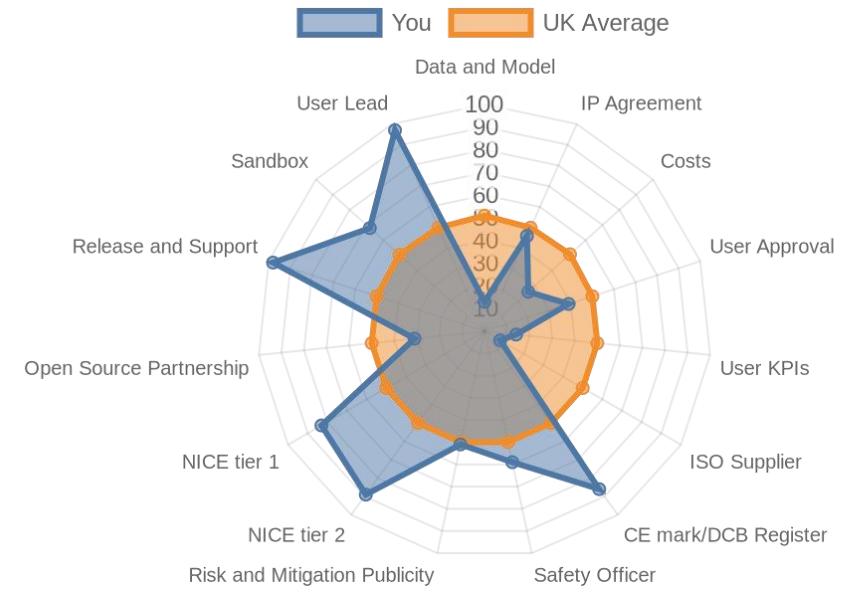
```
Request URL
POST https://quickchart.io/chart/create

Headers Body Params

Header Assignments
Content-Type
VALUE application/json

Transform into Variables
response.url
APPLY TO url

Headers Body Params
Form Data URL Encoded Raw
1 1. {
2 2.   "chart": {
3 3.     "type": "radar",
4 4.     "data": {
5 5.       "labels": {graph_labels},
6 6.       "datasets": [
7 7.         {
8 8.           "label": "You",
9 9.           "data": {user_data}
10 10.        },
11 11.        {
12 12.           "label": "UK Average",
13 13.           "data": {average_data}
14 14.        }]
15 15.     },
16 16.     "options" : {
17 17.       "scale" : {
18 18.         "ticks" : {
19 19.           "min" : 0,
20 20.           "max" : 100,
21 21.           "stepSize" : 10
22 22.         }
23 23.       }
24 24.     }
25 25.   }
```



POST Request

- Add “Content-Type” header with value “application/json”
- Specify the graph in the url body with raw formatting
- Save “response.url” to variable
- **Upon clicking the url, the graph begins to be rendered**
- Note: may not show instantly for complicated graphs

Request URL
POST https://quickchart.io/chart/create

Headers Body Params

Header Assignments

Content-Type

VALUE application/json

Transform into Variables

response.url

APPLY TO url

Headers Body Params

Form Data URL Encoded Raw

```
1  {
2    "chart": {
3      "type": "radar",
4      "data": {
5        "labels": {graph_labels},
6        "datasets": [
7          {
8            "label": "You",
9            "data": {user_data}
10           },
11          {
12            "label": "UK Average",
13            "data": {average_data}
14           ]
15         },
16         "options" : {
17           "scale" : {
18             "ticks" : {
19               "min" : 0,
20               "max" : 100,
21               "stepSize" : 10
22             }
23           }
24         }
25     }
26   }
```

Legend: You (Blue), UK Average (Orange)

Metrics (Axes): Data and Model, IP Agreement, Costs, User Approval, User KPIs, ISO Supplier, CE mark/DCB Register, Safety Officer, Risk and Mitigation Publicity, NICE tier 2

Scale: 0 to 100

Metric	You (Blue)	UK Average (Orange)
Data and Model	100	55
IP Agreement	90	45
Costs	85	40
User Approval	80	35
User KPIs	75	30
ISO Supplier	70	25
CE mark/DCB Register	65	20
Safety Officer	60	15
Risk and Mitigation Publicity	55	10
NICE tier 2	50	10

POST Request

- Add “Content-Type” header with value “application/json”
- Specify the graph in the url body with raw formatting
- Save “response.url” to variable
- Upon clicking the url, the graph begins to be rendered
- **Note: may not render instantly for complicated graphs**

```
Request URL
POST ▾ https://quickchart.io/chart/create

Headers Body Params

Header Assignments
Content-Type
VALUE application/json

Transform into Variables
response.url
APPLY TO url ▾

Headers Body Params

 Form Data  URL Encoded  Raw

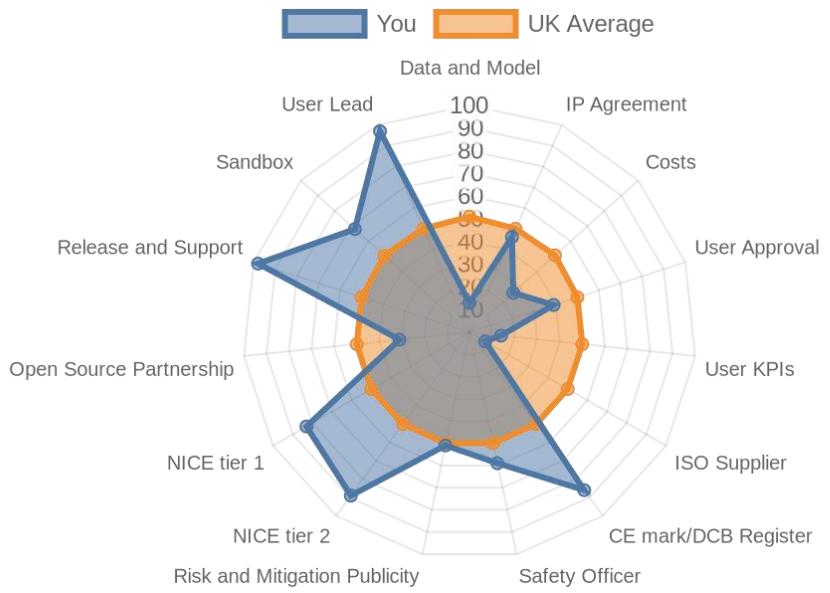


```

1 ▾ {
2 "chart": {
3 "type": "radar",
4 "data": {
5 "labels": {graph_labels},
6 "datasets": [
7 {
8 "label": "You",
9 "data": {user_data}
10 },
11 {
12 "label": "UK Average",
13 "data": {average_data}
14 }
15],
16 "options" : {
17 "scale" : {
18 "ticks" : {
19 "min" : 0,
20 "max" : 100,
21 "stepSize" : 10
22 }
23 }
24 }
25 }
}

```


```



Sendgrid API

POST Request

- Authorization header with private key as value
- Content-Type header with application/json as value
- Body contains sender and recipient email addresses, as well as subject and content of email

Header Assignments

Authorization

VALUE Bearer SG.60RLlqTHRWu_k-UKdeEnGg.r8'

Content-Type

VALUE application/json

Headers (2) Body Params

Form Data URL Encoded Raw

```
1 {  
2   "personalizations": [  
3     {  
4       "to": [  
5         {"email": "{accountEmail}"}  
6       ]  
7     }  
8   ],  
9   "from": {  
10     "email": "fraser.thomson.20@ucl.ac.uk"  
11   },  
12   "subject": "Your email verification code for PRI  
13   "content": [  
14     {  
15       "type": "text/plain",  
16       "value": "{ranint}"  
17     }  
18   ]  
}
```

POST Request

- **Authorization header with private key as value**
- Content-Type header with application/json as value
- Body contains sender and recipient email addresses, as well as subject and content of email

The screenshot shows a configuration interface for a POST request. At the top, there is a section for "Header Assignments" with a red box highlighting the "Authorization" header and its value "Bearer SG.60RLlqTHRWu_k-UKdeEnGg.r8". Below this, there are sections for "Content-Type" and "application/json". The interface includes tabs for "Headers (2)", "Body", and "Params", with "Body" selected. Under "Body", there are radio buttons for "Form Data", "URL Encoded", and "Raw", with "Raw" selected. The JSON content in the "Raw" editor is as follows:

```
1 {  
2   "personalizations": [  
3     {  
4       "to": [  
5         {"email": "{accountEmail}"}  
6       ]  
7     }  
8   ],  
9   "from": {  
10     "email": "fraser.thomson.20@ucl.ac.uk"  
11   },  
12   "subject": "Your email verification code for PRI  
13   "content": [  
14     {  
15       "type": "text/plain",  
16       "value": "{ranint}"  
17     }  
18   ]  
19 }
```

POST Request

- Authorization header with private key as value
- **Content-Type header with application/json as value**
- Body contains sender and recipient email addresses, as well as subject and content of email

Header Assignments

Authorization

VALUE Bearer SG.60RLlqTHRWu_k-UKdeEnGg.r8'

Content-Type

VALUE application/json

Headers (2) Body Params

Form Data URL Encoded Raw

```
1 {  
2   "personalizations": [  
3     {  
4       "to": [  
5         {"email": "{accountEmail}"}  
6       ]  
7     }  
8   ],  
9   "from": {  
10     "email": "fraser.thomson.20@ucl.ac.uk"  
11   },  
12   "subject": "Your email verification code for PRI  
13   "content": [  
14     {  
15       "type": "text/plain",  
16       "value": "{ranint}"  
17     }  
18   ]  
}
```

POST Request

- Authorization header with private key as value
- Content-Type header with application/json as value
- **Body contains sender and recipient email addresses, as well as subject and content of email**

Header Assignments

Authorization

VALUE Bearer SG.60RLlqTHRWu_k-UKdeEnGg.r8'

Content-Type

VALUE application/json

Headers (2) Body Params

Form Data URL Encoded Raw

```
1 {  
2   "personalizations": [  
3     {  
4       "to": [  
5         {"email": "{accountEmail}"}  
6       ]  
7     }  
8   ],  
9   "from": {  
10     "email": "fraser.thomson.20@ucl.ac.uk"  
11   },  
12   "subject": "Your email verification code for PRI  
13   "content": [  
14     {  
15       "type": "text/plain",  
16       "value": "{ranint}"  
17     }  
18   ]  
}
```

Telegram Deployment

Voiceflow API

- POST requests to Voiceflow require secret key and special ID for every user to track state
- Body of the POST request contains the action type (text, choice etc.) and the payload.
- Launch method posts request with action type "launch" to retrieve messages upon launching the chatbot
- Sendmsg method posts request with corresponding action type and the payload

```
1  import telebot
2  import requests
3
4  BOT_API_KEY = "5290147239:AAH05Xle7HJBhoc6Gkyj_usedttJqfLYSjs"
5  VF_API_KEY = "VF.DM.6201623cd72337001b7129d3.9a90ShFO3cbAhamm"
6  bot = telebot.TeleBot(BOT_API_KEY)
7  request_body = ""
8  choice = False
9  choices = []
10 response = ""
11
12 def launch(message):
13     body = {"action": {"type": "launch"}}
14     response = requests.post(
15         f"https://general-runtime.voiceflow.com/state/user/{str(message.chat.id)}/interact",
16         json=body,
17         headers={"Authorization": VF_API_KEY},
18     )
19     return response.json()
20
21 def sendMsg(message):
22     global request_body
23     if request_body == "":
24         request_body = {"action": {"type": "text", "payload": message.text}}
25     response = requests.post(
26         f"https://general-runtime.voiceflow.com/state/user/{str(message.chat.id)}/interact",
27         json=request_body,
28         headers={"Authorization": VF_API_KEY},
29     )
30     return response.json()
31
```

Telegram Deployment

- Uses the telebot library to handle API calls
- TeleBot object created using secret key generated
- Bot.polling() method allows the bot to continuously listen for new messages from user
- Start method is called when user uses the start command in telegram, calls the launch method and outputs the response to the user
- All other messages are passed through handle_message method where the sendMsg method is called with user's input, depending on the response type different action is taken
- Upon type "text" the response is outputted to the user
- Upon type "choice" the choices are stored and displayed to the user, the choice flag is set to true so that the next user response is set as choice type in the voiceflow API call
- Upon type "end" the conversation is terminated.

```
import telebot
import requests

BOT_API_KEY = "5290147239:AAH05Xle7HJBhoc6Gkyj_usedttJqfLYSjs"
VF_API_KEY = "VF.DM.6201623cd72337001b7129d3.9a90ShFO3cbAhamm"
bot = telebot.TeleBot(BOT_API_KEY)
request_body = ""
choice = False
choices = []
response = ""

32 @bot.message_handler(commands=["start"])
33 def start(message):
34     response = launch(message)
35     bot.send_message(message.chat.id, response[-1]["payload"]["message"])
36
37 @bot.message_handler()
38 def handle_message(message):
39     global choice
40     global choices
41     global request_body
42     global response
43     if choice:
44         for c in choices:
45             if c["name"] == message.text:
46                 request_body = c
47     response = sendMsg(message)
48     choice = False
49     choices = []
50     request_body = ""

51
52 for resp in response:
53     if resp["type"] == "text":
54         bot.send_message(message.chat.id, resp["payload"]["message"])
55     elif resp["type"] == "choice":
56         choice = True
57         for choice in resp["payload"]["buttons"]:
58             choices.append(choice)
59             bot.send_message(message.chat.id, choice["name"])
60     elif resp["type"] == "end":
61         bot.send_message(message.chat.id, "Thanks for using our chatbot.")

62
63
64
65 bot.polling()
```

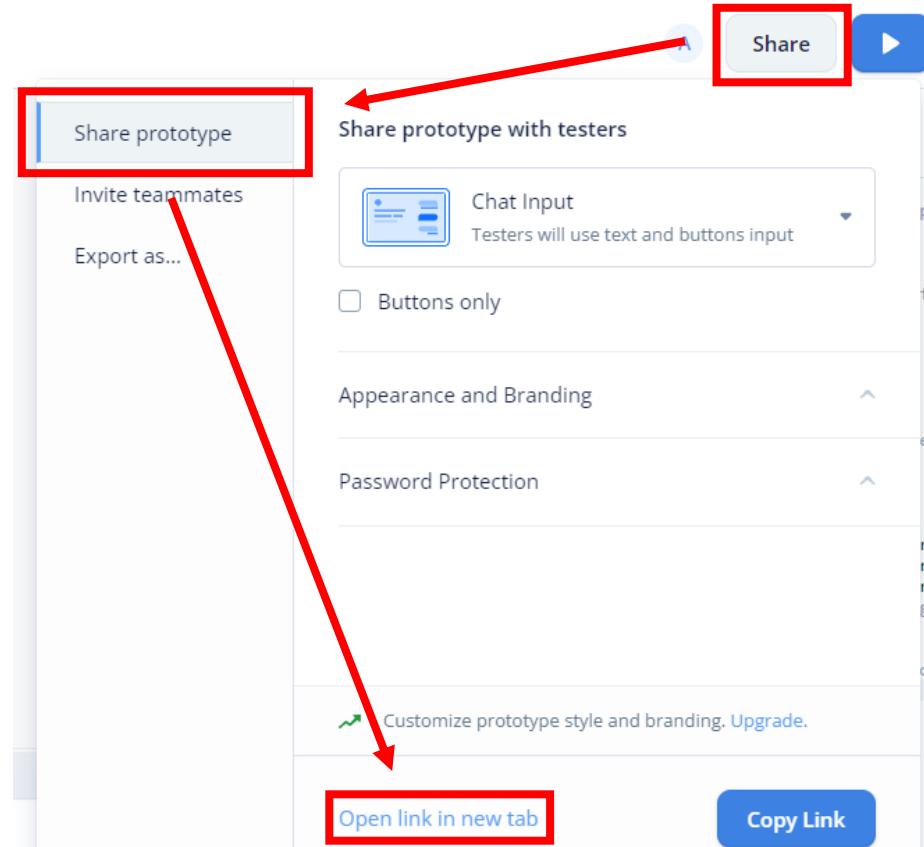
Testing

Testing

- Share prototype as a link
- Traverse down desired path until either hitting an error or natural end of program (closing or refreshing halfway won't work)
- Open “Transcripts” tab
- Make sure “Debug Messages” is ticked

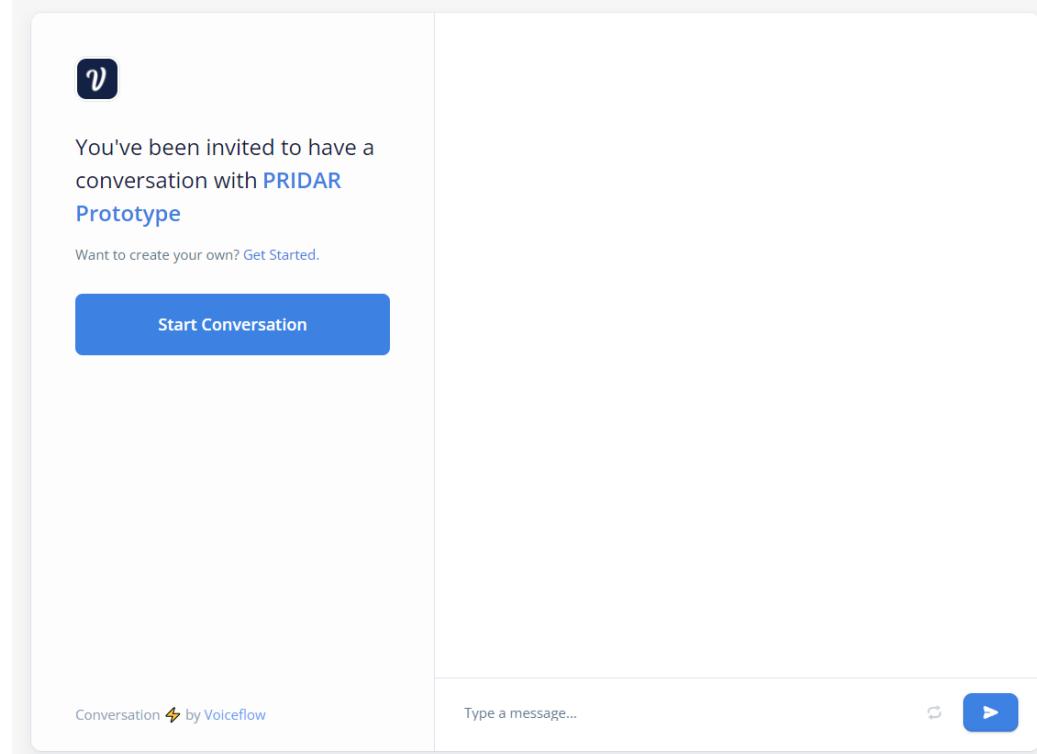
Testing

- **Share prototype as a link**
- Traverse down desired path until either hitting an error or natural end of program (closing or refreshing halfway won't work)
- Open “Transcripts” tab
- Make sure “Debug Messages” is ticked



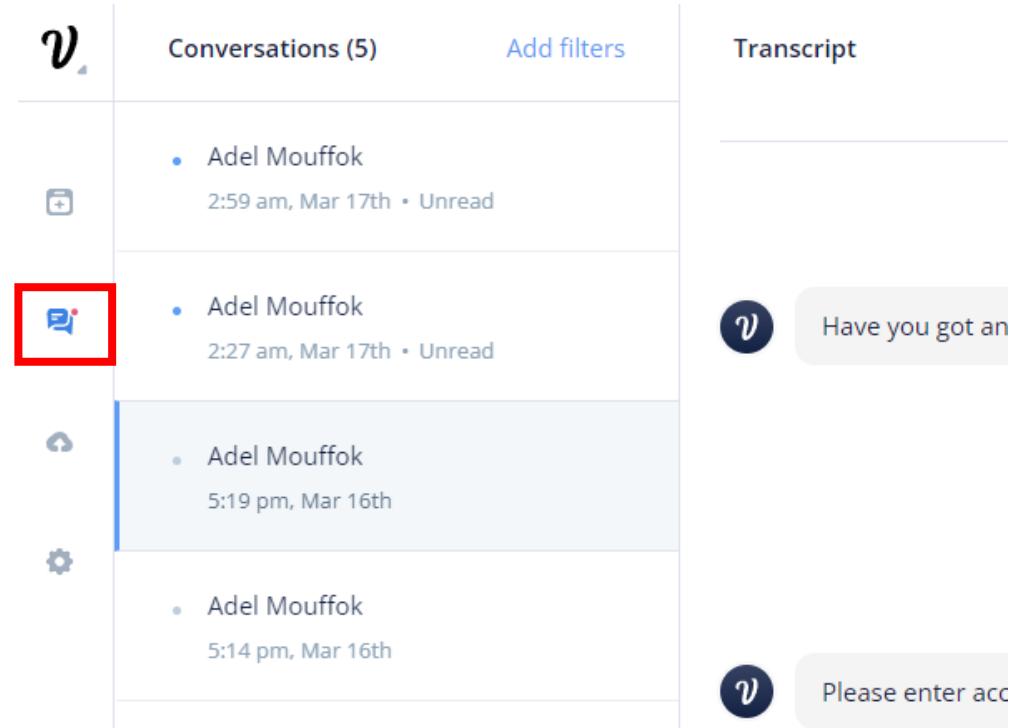
Testing

- Share prototype as a link
- **Traverse down desired path until either hitting an error or natural end of program (closing or refreshing halfway won't work)**
- Open “Transcripts” tab
- Make sure “Debug Messages” is ticked



Testing

- Share prototype as a link
- Traverse down desired path until either hitting an error or natural end of program (closing or refreshing halfway won't work)
- **Open “Transcripts” tab**
- Make sure “Debug Messages” is ticked



Testing

- Share prototype as a link
- Traverse down desired path until either hitting an error or natural end of program (closing or refreshing halfway won't work)
- Open “Transcripts” tab
- **Make sure “Debug Messages” is ticked**

