Citizen Science Survey is a part of two projects

1. Qualtrics Automation
2. Citizen Science Backend

1. Project 1 → Qualtrics Automation

Requirement →

Create a Qualtrics survey from csv documents.

There are two documents needed in order to compile the survey→

1. Documents consisting of survey questions (e.g. Hate Speech in the first task) e.g. citizen_science_survey.csv
2. Questions and options related to each question. e.g./src/data/hate_speech_question.csv and /src/data/hate_speech_question2.csv

Process

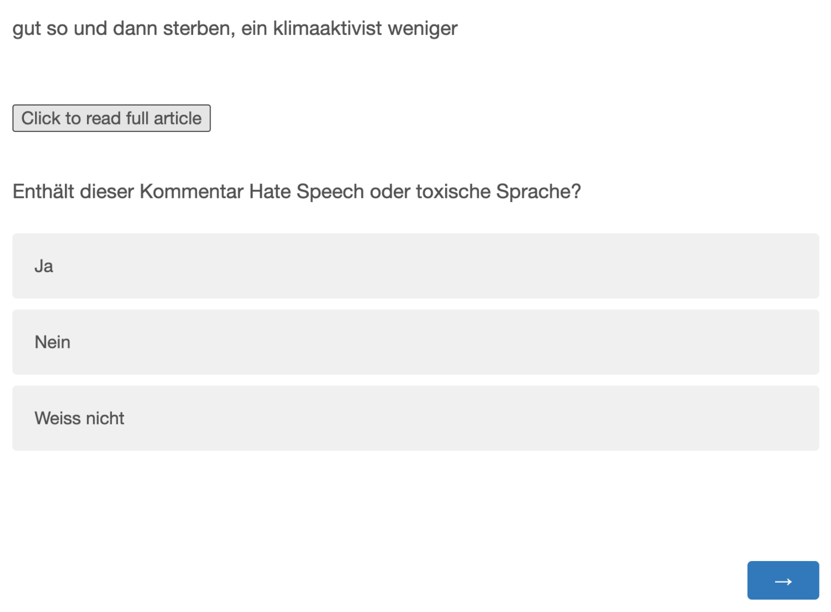
First document should be specified in the file -

/src/APIs/Automate_qualtrics.py

```
for filename in os.listdir(full_path):
   if (filename=="citizen_science_survey.csv"):
       file_name = filename
       survey_name = filename.split('.')[0]
       survey_data['SurveyName'] = survey_name
```

Automate_qualtrics.py is executed to create the survey. It internally calls Qualtrics API for this purpose. These APIs like creating survey, adding questions need data in a particular json format. Thus Automate_qualtrics.py file provides these json in required format while calling these APIs.

To fulfill the format of json file, we are using pre defined json files. These files have been created before the experiment as per our requirement. If we need a block like below where we have our first question only as a text and another question as a multiple choice option —>



gut so und dann sterben, ein klimaaktivist weniger

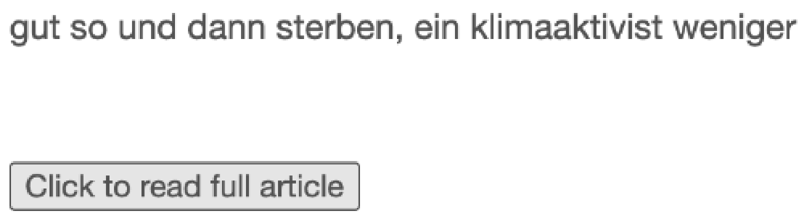
Enthält dieser Kommentar Hate Speech oder toxische Sprache?

Figure showing a block with two questions.

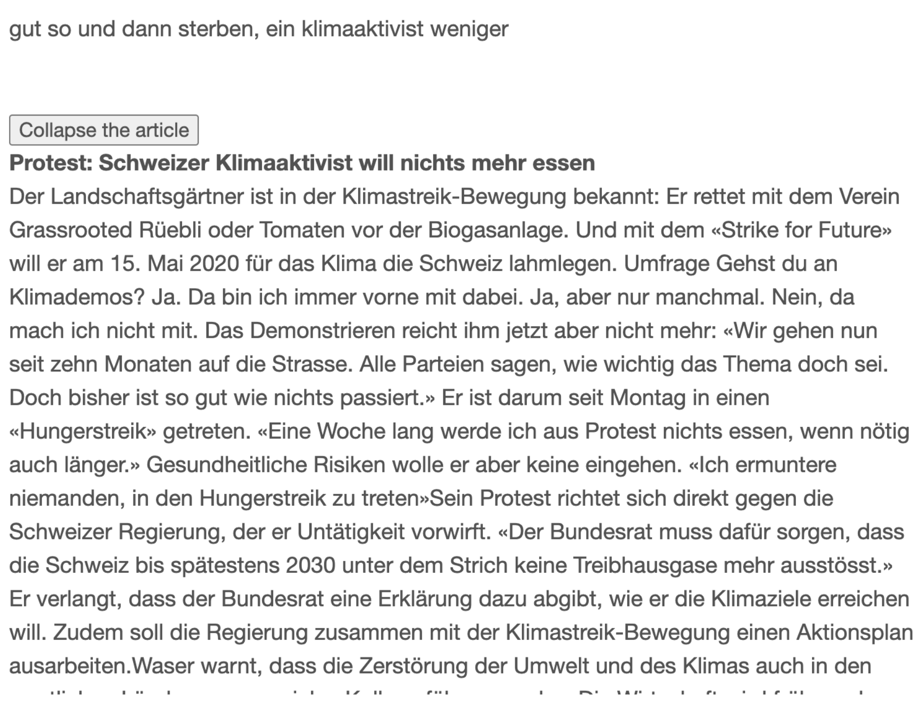
We have used two json file for the creation of above format

src/data/question_1.json and src/data/question.json

As you can see, Question_1.json doesn't contain any option field. Its to generate below format as shown in next two images.



gut so und dann sterben, ein klimaaktivist weniger


gut so und dann sterben, ein klimaaktivist weniger

Protest: Schweizer Klimaaktivist will nichts mehr essen
Der Landschaftsgärtner ist in der Klimastreik-Bewegung bekannt: Er rettet mit dem Verein Grassrooted Rüebli oder Tomaten vor der Biogasanlage. Und mit dem «Strike for Future» will er am 15. Mai 2020 für das Klima die Schweiz lahmlegen. Umfrage Gehst du an Klimademos? Ja. Da bin ich immer vorne mit dabei. Ja, aber nur manchmal. Nein, da mach ich nicht mit. Das Demonstrieren reicht ihm jetzt aber nicht mehr: «Wir gehen nun seit zehn Monaten auf die Strasse. Alle Parteien sagen, wie wichtig das Thema doch sei. Doch bisher ist so gut wie nichts passiert.» Er ist darum seit Montag in einen «Hungerstreik» getreten. «Eine Woche lang werde ich aus Protest nichts essen, wenn nötig auch länger.» Gesundheitliche Risiken wolle er aber keine eingehen. «Ich ermuntere niemanden, in den Hungerstreik zu treten»Sein Protest richtet sich direkt gegen die Schweizer Regierung, der er Untätigkeit vorwirft. «Der Bundesrat muss dafür sorgen, dass die Schweiz bis spätestens 2030 unter dem Strich keine Treibhausgase mehr ausstösst.» Er verlangt, dass der Bundesrat eine Erklärung dazu abgibt, wie er die Klimaziele erreichen will. Zudem soll die Regierung zusammen mit der Klimastreik-Bewegung einen Aktionsplan ausarbeiten.Waser warnt, dass die Zerstörung der Umwelt und des Klimas auch in den

The below json supports text questions shown above

/src/data/question_1.json

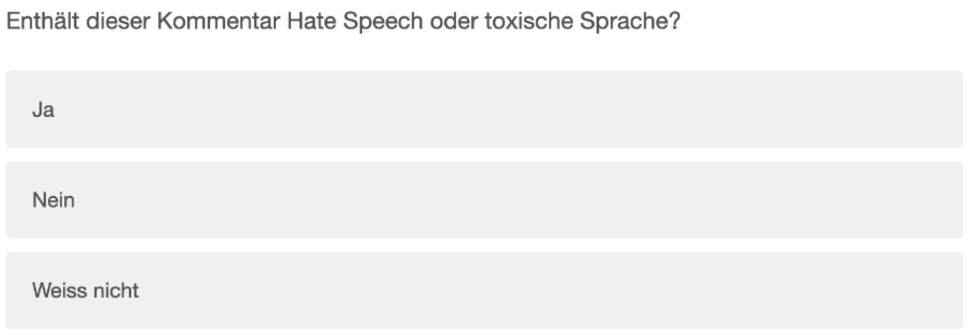
```
{
   "Configuration":{
       "QuestionDescriptionOption": "UseText"
   },
   "Language":[
   ],
   "NextChoiceId": 4,
   "NextAnswerId": 1,
   "QuestionID": "Q4",
   "QuestionDescription":"This is my second block test question",
   "QuestionText":"This is my second block test question <div><br></div>",
   "QuestionType":"MC",
   "Selector":"SAVR",
   "Validation":{
       "Settings":
       {
           "Type": "None"
       }
   },
   "QuestionJS":""


}
```

question_1.json

Here question_1.json provides us data in the format for the first question which is only text field. You can see clearly there is no option field here.

Where as src/data/question.json contains options field in order to generate below question as shown in below images-


Enthält dieser Kommentar Hate Speech oder toxische Sprache?

We need this format of json file to generate above question.

```
{
    "ChoiceOrder":[
        "1",
        "2",
        "3"
    ],
    "Choices":{
        "1":{
            "Display":"Choice no 1 is this"
        },
        "2":{
            "Display":"Choice no 2 is that"
        },
        "3":{
            "Display":"Choice no 3 is what"
        }
    },
    "Configuration":{
        "QuestionDescriptionOption": "UseText"
    },
    "Language":[
    ],
    "NextChoiceId": 4,
    "NextAnswerId": 1,
    "QuestionID": "Q4",
    "QuestionDescription":"This is my second block test question",
    "QuestionText":"This is my second block test question <div><br></div>",
    "QuestionType":"MC",
    "Selector":"SAVR",
    "SubSelector":"TX",
    "Validation":{
        "Settings":
        {
            "Type": "None"
        }
    },
    "QuestionJS":""


}
```

As you can question.json provides us format for the second question which include three options.

Populating json file

Currently, question.json and question_1.json contains the placeholder data only. This data is meant to be modified in the program.

For text question(without any comments), we use below code to populate the data-

```
question_json1['QuestionDescription'] = row['Kommentar']
question_json1['QuestionText'] =
row['Kommentar']+question_kommentar_titel_js+row["Titel"]+question_titel_text_js+row["
Text"]+question_text_end_js
question_json1["DataExportTag"] = "Comment "+str(block_number)
question_json1["QuestionType"] = "DB"
question_json1["Selector"] = "TB"
question_json1["QuestionJS"] = js_script
```

We first read ***document 1*** and iterate over each row to populate the json.

Below code shows we open the ***document2*** in order to copy the data into json

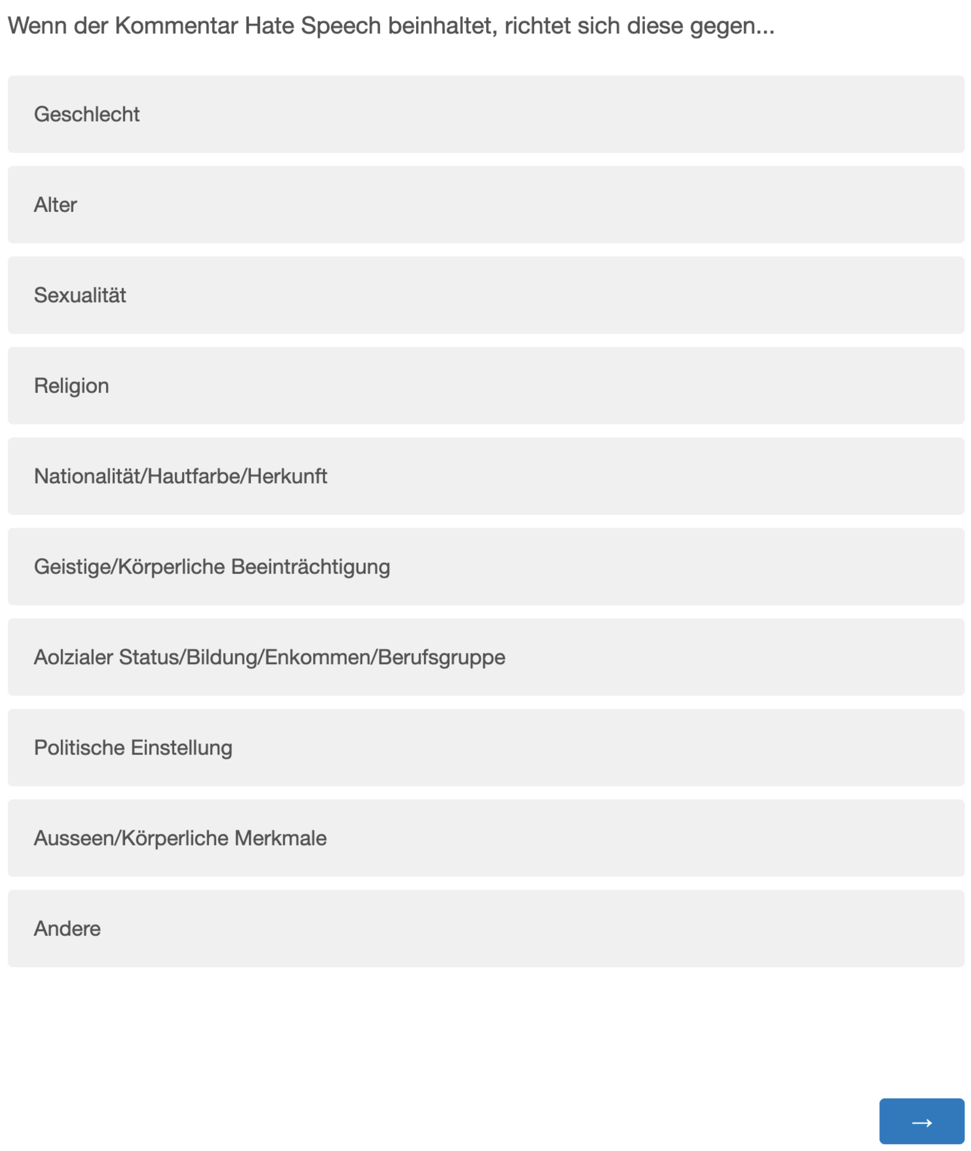
```
       with open(full_path+"/"+hate_speech_file, mode='r') as hs_file:
           hs_csv_reader = csv.DictReader(hs_file)
           for row in hs_csv_reader:
                hate_speech_json['QuestionDescription'] = row["Question Text"]
                hate_speech_json['QuestionText'] = row["Question Text"]
                hate_speech_json["Choices"]["1"]['Display'] = row['Option 1']
#Uncomment if you are passing block from file.
                hate_speech_json["Choices"]["2"]['Display'] = row['Option 2']
#Uncomment if you are passing block from file.
                hate_speech_json["Choices"]["3"]['Display'] = row['Option 3']
#Uncomment if you are passing block from file.
                hate_speech_json["QuestionID"] = row['Question'] #Uncomment if you are
passing block from file.
                hate_speech_json["QuestionJS"] = js_script_hs
```

The data to be populated is extracted from document 1 and 2 combined for one block.

Current survey required us to have one Comment (which is actually divided into three qualtrics questions).

We have seen how first two questions are created.

The third question is created using another json file. This question is supposed to look like this -



Wenn der Kommentar Hate Speech beinhaltet, richtet sich diese gegen...

Geschlecht

Alter

Sexualität

Religion

Nationalität/Hautfarbe/Herkunft

Geistige/Körperliche Beeinträchtigung

Aolzialer Status/Bildung/Enkommen/Berufsgruppe

Politische Einstellung

Ausseen/Körperliche Merkmale

Andere

Below code shows the json file →

```
{
   "ChoiceOrder":[
       "1",
       "2",
       "3",
       "4",
       "5",
       "6",
       "7",
       "8",
       "9",
       "10"
   ],
   "Choices":{
       "1":{
           "Display":"Choice no 1 is this"
       },
       "2":{
           "Display":"Choice no 2 is that"
       },
       "3":{
           "Display":"Choice no 3 is what"
       },
       "4":{
           "Display":"Choice no 1 is this"
       },
       "5":{
           "Display":"Choice no 2 is that"
       },
       "6":{
           "Display":"Choice no 3 is what"
       },
       "7":{
           "Display":"Choice no 1 is this"
       },
       "8":{
           "Display":"Choice no 2 is that"
       },
       "9":{
           "Display":"Choice no 3 is what"
       },
       "10":{
           "Display":"Choice no 3 is what"
       }
   },
   "Configuration":{
       "QuestionDescriptionOption": "UseText"
   },
   "Language":[
   ],
   "NextChoiceId": 4,
   "NextAnswerId": 1,
   "QuestionID": "Q4",
   "QuestionDescription":"This is my second block test question",
   "QuestionText":"This is my second block test question <div><br></div>",
   "QuestionType":"MC",
   "Selector":"MAVR",
   "SubSelector":"TX",
   "Validation":{
           "Settings":
           {
               "ForceResponse": "OFF",
               "Type": "None"
           }
   },
   "QuestionJS":"",
   "DisplayLogic": {
       "0": {
           "0": {
               "ChoiceLocator": "q://QID16/SelectableChoice/1",
               "Description": "<span class=\"ConjDesc\">If</span> <span
class=\"QuestionDesc\">Enthält dieser Kommentar Hate Speech oder toxische
Sprache?</span> <span class=\"LeftOpDesc\">Ja</span> <span class=\"OpDesc\">Is
Selected</span> ",
               "LeftOperand": "q://QID16/SelectableChoice/1",
               "LogicType": "Question",
               "Operator": "Selected",
               "QuestionID": "QID16",
               "QuestionIDFromLocator": "QID16",
               "QuestionIsInLoop": "no",
               "Type": "Expression"
           },
           "Type": "If"
       },
       "Type": "BooleanExpression",
       "inPage": false
   }


}
```

question_hs.json

This json needs additional field to show conditional display.

Below code populates the data from document_2 into question_hs.json

```
with open(full_path+"/"+hate_speech_file_2, mode='r') as hs_file_2:
           hs_csv_reader = csv.DictReader(hs_file_2)
           for row in hs_csv_reader:
                hate_speech_json_2['QuestionDescription'] = row["Question Text"]
                hate_speech_json_2['QuestionText'] = row["Question Text"]
                hate_speech_json_2["Choices"]["1"]['Display'] = row['Option 1']
#Uncomment if you are passing block from file.
                hate_speech_json_2["Choices"]["2"]['Display'] = row['Option 2']
#Uncomment if you are passing block from file.
                hate_speech_json_2["Choices"]["3"]['Display'] = row['Option 3']
                hate_speech_json_2["Choices"]["4"]['Display'] = row['Option 4']
#Uncomment if you are passing block from file.
                hate_speech_json_2["Choices"]["5"]['Display'] = row['Option 5']
#Uncomment if you are passing block from file.
                hate_speech_json_2["Choices"]["6"]['Display'] = row['Option 6']
                hate_speech_json_2["Choices"]["7"]['Display'] = row['Option 7']
#Uncomment if you are passing block from file.
                hate_speech_json_2["Choices"]["8"]['Display'] = row['Option 8']
#Uncomment if you are passing block from file.
                hate_speech_json_2["Choices"]["9"]['Display'] = row['Option 9']
                hate_speech_json_2["Choices"]["10"]['Display'] = row['Option 10']
                hate_speech_json_2["QuestionID"] = row['Question']
                hate_speech_json_2["QuestionJS"] = js_script_h
```

After changing the code and json as per need, please run automate_qualtrcis.py

Project 2 → Citizen Science Backend

Citizen Science backend handles assigning unique question set to the connected clients. The current settings helps to duplicate each question 3 times and assign 30 questions to a user and for a total of 50 users.

In order to make this work, we need to run serially→

1. ***src/backend/random_generator.py*** → This file generates a list of 30 random questions for each of the 50 users. The generated data is stored in file → *src/backend/data/question_list.json*

2. ***src/backend/create_db_entry.py*** → This file is responsible to upload the data into mongodb Database.

Important point: Before running this program make sure that mongo db docker container is running on the server. Use below command to run it →

```
docker run -d -p 27017:27017 --name prodigi-mongo mongo:latest
```

So far you can run these two files in your local system also.

But you need to start Flask and WSGI server to enable the API endpoint by which qualtrics client can connect and obtain the question list. This API endpoint is →

```
/api/v1/questions
```

To run the Flask server in our Ubuntu server, we have a service which automatically runs the flask server. Please make sure that the service is active. The service automatically starts upon server restart. Check for error logs in case its not running.

/etc/systemd/system → citizen_backend.service

Revers proxy to the flask server has been implemented using Nginx. You can find more details about the file on the server.

/etc/nginx/sites-available → citizen-science.org