# Installation

## Dependencies

1. Install the following dependencies
   1. **Tessaract-OCR**: https://digi.bib.uni-mannheim.de/tesseract/tesseract-ocr-w64-setup-v5.0.0-alpha.20210506.exe
   2. **Poppler**: https://blog.alivate.com.au/poppler-windows/
   3. **Git**: https://git-scm.com/download/win
   4. **MS Office**
2. Once the above dependencies are installed, add the path to System Environment Variables path parameters. Sample paths added are shown below. Paths may vary depending on the installation directory.
   1. C:\Program Files (x86)\Tesseract-OCR
   2. C:\Program Files\poppler-0.68.0\bin
   3. C:\Program Files\Git\cmd
   4. C:\Program Files\Git\bin
   5. C:\Program Files\Git\bin\git.exe

## Python packages

1. All the required python packages are listed in requirements.txt file.
2. Create a virtual environment and activate it
   1. pip install virtualenv
   2. python -m venv amplelogic\_env
   3. .\ amplelogic\_env \Scripts\activate
3. Install all the required packages using below code
   1. pip install -r requirements.txt
   2. pip install git+https://github.com/boudinfl/pke.git

## One time dependencies

Do the following whenever the application is run in the newly created environment

1. Run the following command in shell (command prompt)
   1. python -m spacy download en\_core\_web\_sm
2. Run the following in python environment
   1. import nltk
   2. nltk.download('brown')
   3. nltk.download('stopwords')
   4. nltk.download('popular')

# Structure

1. All the python files are saved in the same folder along with s2v\_old
2. Create three new folders inside the main directory
   1. Input: All the input files (PDF, docx, pptx, base64 as txt) must be saved here
   2. Output: All the output files are saved by the application in this folder. The output is saved as an .xlsx file with one sheet for MCQ questions and another sheet for True/False questions.
   3. temp\_files: This is an empty folder. All the intermediate files are stored in this folder and once the application is run successfully, then all the contents inside this folder are deleted automatically by the application.
3. Output format: The .xlsx file stored in Output folder has the following format. It contains two sheets.
   1. MCQ Questions: It contains SNo, Question, OptionA, OptionB, OptionC, OptionD, Answer, Question Context
   2. TrueFalse Questions: It contains SNo, Question, Answer, Question Context

# File formats

1. The application supports the following file formats
   1. PDFs with images
   2. PDFs as text
   3. docx
   4. pptx
   5. PDFs converted in base64 format as txt
2. It is assumed that only PDF with images have watermark and users will correctly specify if the document contains watermark.
3. The application does not support if the images inside PDF are tilted/skewed.
4. All text files are base64 format of only PDF files.

# Application

To use the question generation application the server must be started. As all the API’s are generated using FastAPI, the server used here is uvicorn.

1. Use the below command to start the server
   1. uvicorn main:app –host <host number> --port <port number>
   2. When run for the first time, the server automatically downloads all required files.
   3. Once the server is up, the application can be tested using either postman or swagger.
2. The API links are
   1. http://<host-name>:<port-number>/get\_paths.

This takes three input parameters - input\_file\_name\_1, input\_file\_name\_2, contains\_watermark. input\_file\_name\_1 is required field, contains\_watermark is a boolean field.

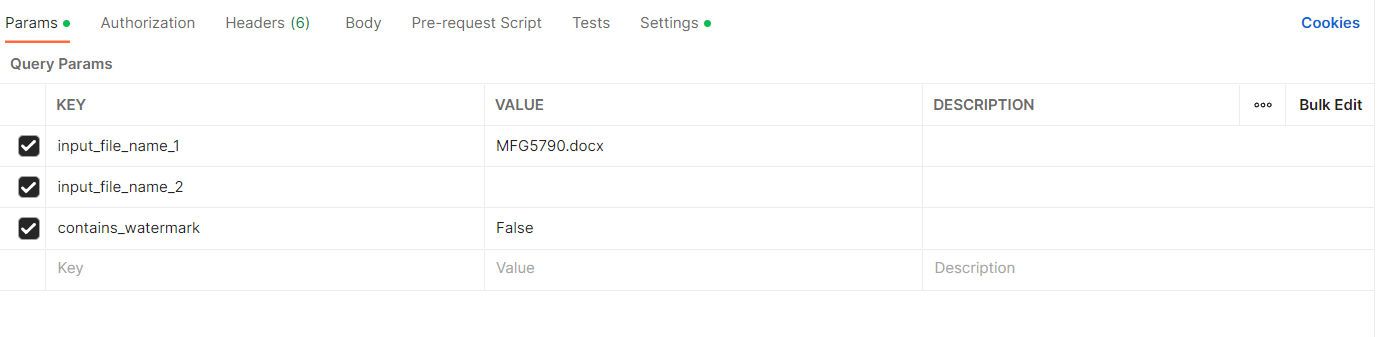
* 1. http://<host-name>:<port-number>/ sections\_to\_consider

This takes two input parameters - sections\_to\_consider and sections\_to\_eliminate. Both are optional fields.

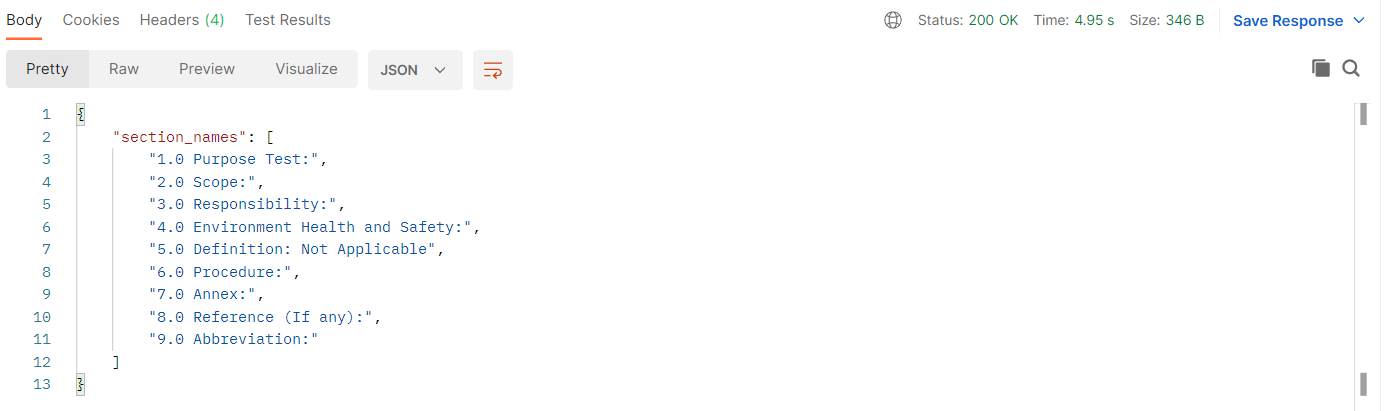
## Running application for new file

For running the question generation application for a new file, ensure that the new file is present in Input directory. Once the server is up,

1. Give the name of the file for the input parameter ‘input\_file\_name\_1’ like MFG5790.docx. Leave ‘input\_file\_name\_2’ field blank. If the input file is a PDF file and if it contains watermark then give ‘contains\_watermark’ as True, otherwise, leave it to False (default).
2. For the application running in local host, the API for the above example looks like http://127.0.0.1:8000/get\_paths?input\_file\_name\_1=MFG5790.docx&input\_file\_name\_2=&contains\_watermark=False



1. The expected output is in json format with section names. Select the section from which the questions to be generated. If the section names could not be extracted then the application sends a message saying the section names have to it chosen manually.



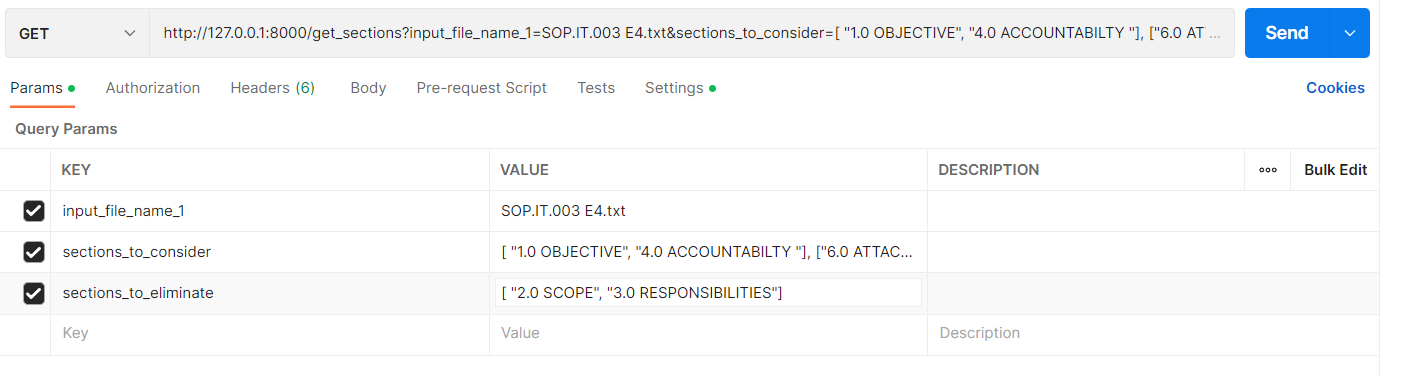
1. If the section names are given manually then do not pass numbers. For example, if the section name is ‘1.0 Objective:’ then enter ‘Objective’.
2. The user then has to input the sections names to be considered or section names to be eliminated in the second API call along with the input\_file\_name\_1.
3. The format for entering the sections is ‘[section\_start\_name, section\_end\_name], [‘section\_start\_name, section\_end\_name],…’. Users can chose any number of sections to either consider or eliminate. Section\_end\_name acts as the end point and that particular section is not considered. It only considers all sections from section\_start\_name (included) till section\_end\_name (excluded).
4. If user wants to consider the all the sections between "1.0 OBJECTIVE", "4.0 ACCOUNTABILTY " and "6.0 ATTACHMENTS", "8.0 ABBREVIATIONS" but exclude "2.0 SCOPE", "3.0 RESPONSIBILITIES" then user can input as shown below in postman.

sections\_to\_consider: [ "1.0 OBJECTIVE", "4.0 ACCOUNTABILTY "], ["6.0 ATTACHMENTS", "8.0 ABBREVIATIONS"]

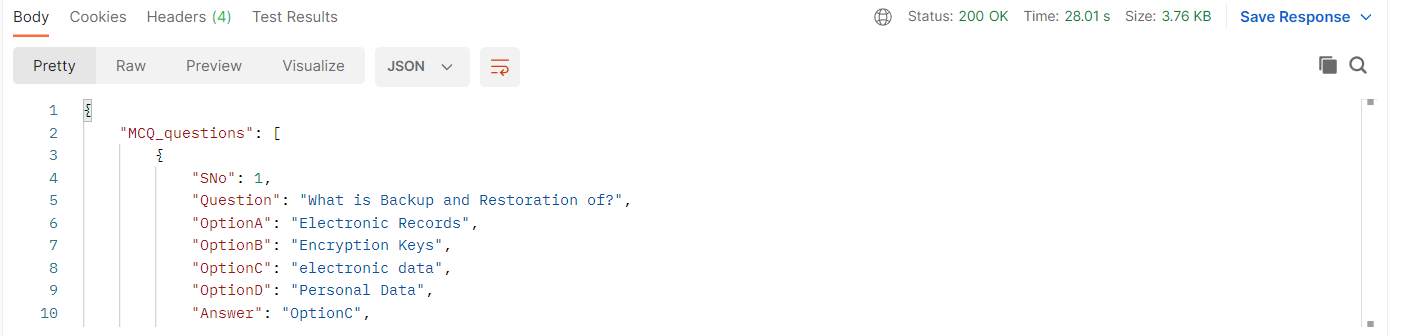
sections\_to\_eliminate: [ "2.0 SCOPE", "3.0 RESPONSIBILITIES"]

1. The API running in localhost for the above example looks like

http://127.0.0.1:8000/get\_sections?sections\_to\_consider=[ "1.0 OBJECTIVE", "4.0 ACCOUNTABILTY "], ["6.0 ATTACHMENTS", "8.0 ABBREVIATIONS"]&sections\_to\_eliminate=[ "2.0 SCOPE", "3.0 RESPONSIBILITIES"]



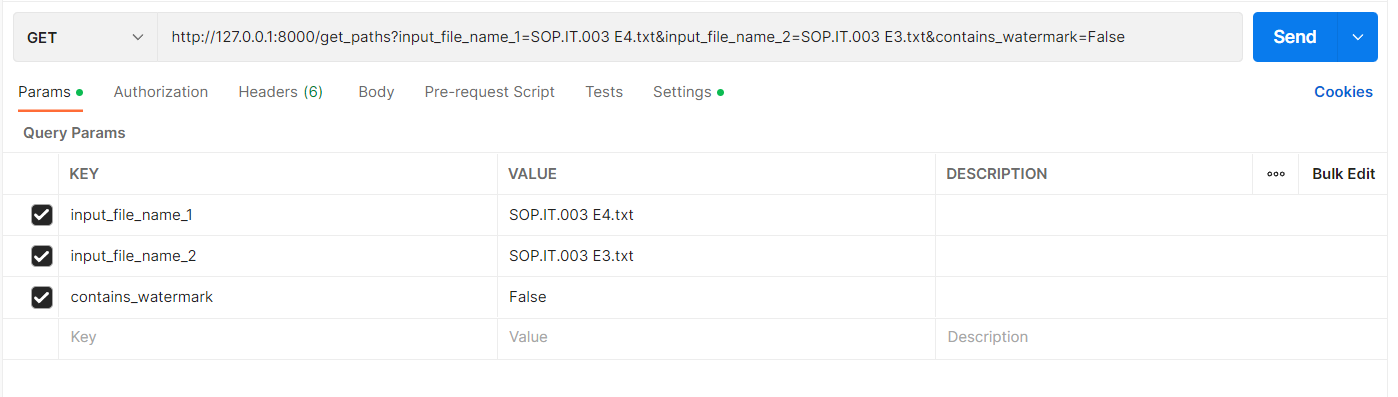
1. The output is saved as .xlsx file in the output folder and the same is returned to the UI in JSON format.



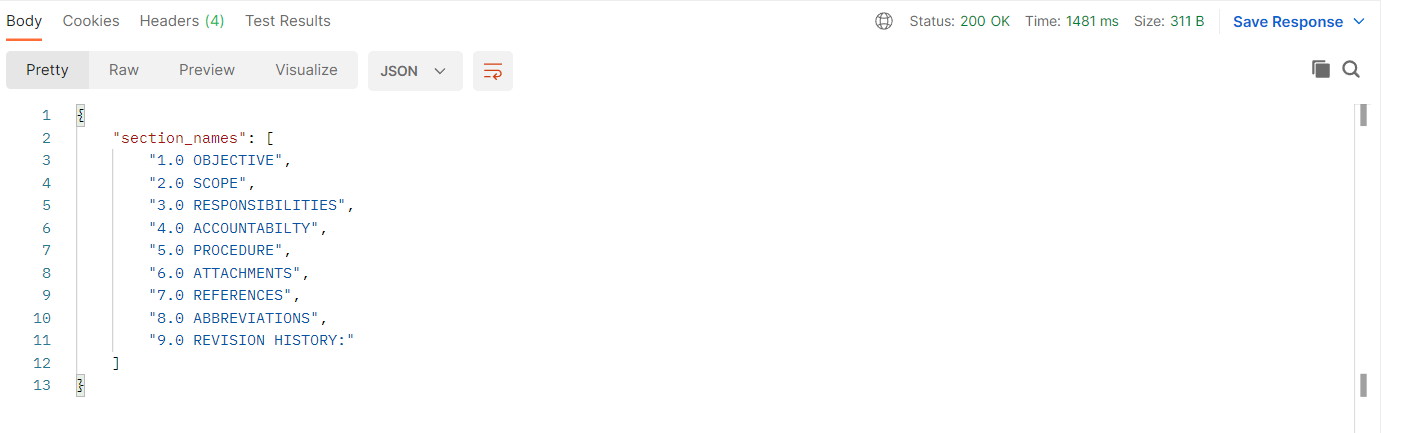
## Running application for updated file

For running the question generation application for a new file, ensure that both the new file and old file are present in the Input directory. Once the server is up,

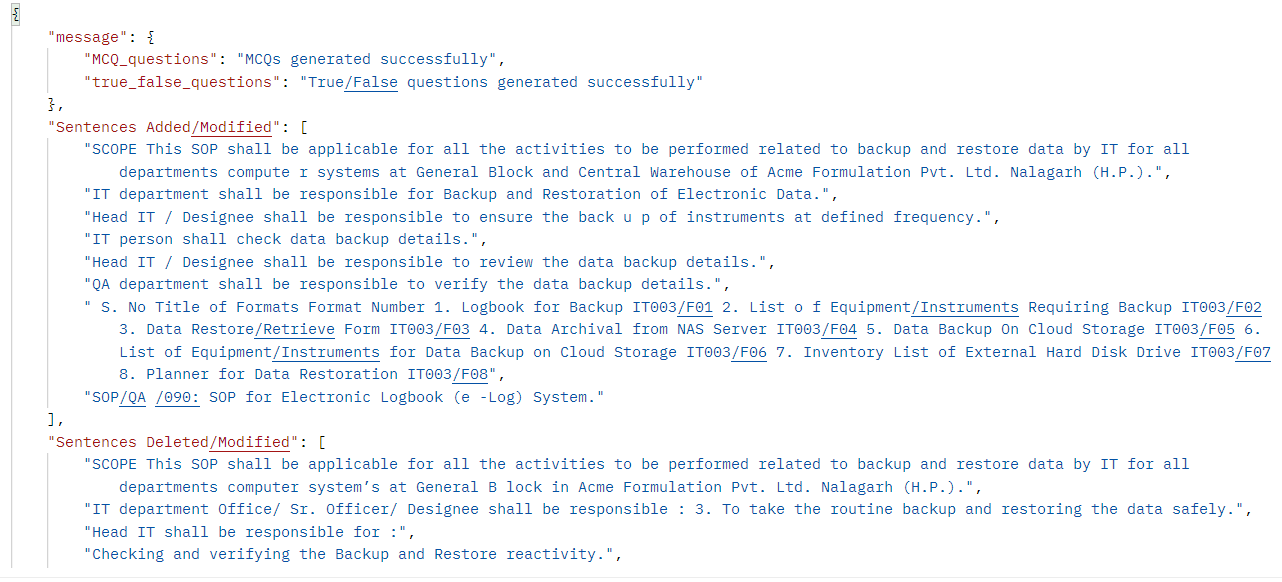
1. Give the name of the file for the input parameter ‘input\_file\_name\_1’ like SOP.IT.003 E4.txt. ‘input\_file\_name\_2’ like SOP.IT.003 E3.txt. If the input file is a PDF file and if it contains watermark then give ‘contains\_watermark’ as True, otherwise, leave it as False (default).
2. Note that input\_file\_name\_1 is the new file and input\_file\_name\_2 is the old file.



1. For the application running in local host, the API for the above example looks like http://127.0.0.1:8000/get\_paths?input\_file\_name\_1=SOP.IT.003 E4.txt&input\_file\_name\_2=SOP.IT.003 E3.txt&contains\_watermark=False



1. The second API input follows the same procedure as for new file.
2. The output is saved as .xlsx file in the output folder and also returned json format. Along with the output, the sections added/modified and deleted/modified are also returned to the user.



## Cleaning the folders

Intermittently, all the files and folders inside ‘temp\_files’ (only the files and folders inside, not the original folder) must be deleted manually. This part was done programmatically before but later modified this functionality to include changes from multiple users.