Team TheThree - Task 1

Allow users to configure the way to mapping the attribute name and its value

Requirements

Write a java/python command line application to parse the input and write to JSON file with configured file.

> Input: `ReqIf file` (same as previous round) and configuration file (can be use single configuration file for task 1 and task 2)

> Output: `JSON file` with mapped content

Evaluation

- `[x]` Array mapping
- `[x]` Attribute name and the same value
- `[x]` Attribute name and customize value
- `[x]` Clean code

Installation

Install the required packages by using `pip`

```
$ pip install -r requirements.txt
```

Getting Started

```
$ python main.py -i Requirements.reqif -o test.json -s config.yml
```

Usage

The program have `3 arguments`. You can check the documentation by using the command

```
$ python main.py -h
```

```
usage: main.py [-h] [-i INPUT_FILE] [-o OUTPUT_FILE]

options:
```

```
  -h, --help            show this help message and exit
  -i INPUT_FILE, --input_file INPUT_FILE
                        Directory to input file. Accepts file *.reqif or
*.xml only
  -o OUTPUT_FILE, --output_file OUTPUT_FILE
                        Directory to output *.json file.
  -s SETTINGS, --settings SETTINGS
                        Directory to configure settings *.yml file
```

Example command:

```
$ python main.py -i Requirements.reqif -o test.json -s config.yml
```

Experiment

> Keep in mind that the input file will not changed. So way to crawl and get the information in the input file will not changed.

In the previous round, we were able to crawl the data from `.reqif` files and export to file json format with hard code mapping `key` and `value` which is contained in `.reqif` file and mapping to new `key` and `value` by rule can extract by looking file sample output `.json`.

In this round, we were able to mapping with definitions in configuring file. We choose to `yaml` format to explore the configuration. Here is our configuration to got the output like as the previous round.

```yaml
module:
  name:
    key: Module Name

  type:
    key: Module Type

  artifacts:
    key: List Artifact Info
    value_type: array
    artifact:
      type:
        key: Attribute Type
      ReqIF.ForeignID:
        key: Identifier
        value_type: number
      ReqIF.ForeignCreatedBy:
        key: Creator
      ReqIF.ForeignModifiedBy:
        key: Contributor
      ReqIF.ForeignCreatedOn:
        key: Created On
        value_type: date
      ReqIF.ForeignModifiedOn:
```

```
        key: Modified On
        value_type: date
      ReqIF.Text:
        value_type: html_string
      ReqIF.Name:
        key: Title
      ReqIF.ChapterName:
        key: ReqIF.Text
        value_type: html_string
      ReqIF.Description:
        key: Description

      Status:
        value_mapping:
          NEW/CHANGED: New/Changed
      Artifact Format:
        ignore: True

     ...
```

For more details about guidelines of the configuration file, see the documentation `Configure.pdf`.

Just looking the configuration for `Task 1`. After parser the configuration we got the rule of array mapping for attribute name and attribute value.

```
 if attr_name in config:
    if config.get(attr_name).get('ignore') == True:
        continue
    set_value(attributes, attr_value,
              config[attr_name], attr_name=attr_name)
```

```
def load_config(filename: str) -> dict:
    """
    Loads the config file and returns the module config
    """
    if is_locked(filename) is not True:
        return yaml.safe_load(open(filename, 'r'))['module']
    else:
        raise KeyError(
            'File %s is locked, please end to write the file and close the
file' % filename)


def map_value(config: dict, value):
    """
    Maps the value to the config value mapping
    """
    if 'value_mapping' in config:
        default_value = config.get('default_value', value)
```

```
            value = config['value_mapping'].get(value, default_value)
        return value


    def format_value(config: dict, value):
        """
        Formats the value according to the config
        """
        value_type = config.get('value_type')

        if isinstance(value, dict) and (value_type is None or value_type ==
    'string'):
            value = value.get('div', {}).get('#text', '')

        if value_type == 'number':
            value = int(value)
        elif value_type == 'html_string':
            value = xmltodict.unparse(value, pretty=True)[39:]

        return value


    def set_value(data: dict, value, config: dict, attr_name='') -> None:
        """
        Sets the value in the data dict according to the config
        """
        value = map_value(config, value)
        value = format_value(config, value)
        data[config.get('key', attr_name)] = value
```

Fixing some problems in previous rounds

In previous rounds, we got some feedback from organizers that we would not check the file is handled or not by other processes before open that file or write that file. So in this round we update condition check that file `is_locked` or not.

```
def is_locked(filename):
    system = platform.system()

    full_path = os.path.abspath(filename)

    if system == 'Linux':
        return is_locked_in_Linux(full_path)
    elif system == 'Windows':
        return is_locked_in_Windows(full_path)
```

- Condition for `Windows OS` system:

```
def is_locked_in_Windows(full_path):
    for proc in psutil.process_iter():
        try:
            for item in proc.open_files():
                if full_path == item.path:
                    return True
        except Exception:
            print(full_path, '-> Failed to open file')

    return False
```

- Condition for `Linux OS` system:

```
def is_file_locked(path):
    try:
        # Open the file in exclusive mode
        with open(path, 'r') as file:
            return False
    except IOError:
        # File is locked by another process
        return True


def is_directory_locked(path):
    try:
        # Attempt to rename the directory
        os.rename(path, path)
        return False
    except OSError:
        # Directory is locked by another process
        return True


def is_locked_in_Linux(path):
    if os.path.isfile(path):
        return is_file_locked(path)
    elif os.path.isdir(path):
        return is_directory_locked(path)
    else:
        print("The specified path does not exist.")
        return None
```

Demo

Here is our result which runs in the sample file `Requirements.reqif` and with configures file `config.yml`.

```
{
  "Module Name": "ECU_Requirement",
  "Module Type": "empty",
  "List Artifact Info": [
    {
      "Attribute Type": "Heading",
      "Contributor": "kit7fe",
      "Created On": "2019-10-08T06:18:45.677Z",
      "Creator": "kit7fe",
      "Description": "",
      "Identifier": 629021,
      "Modified On": "2019-10-08T06:18:45.677Z",
      "ReqIF.Text": "<div
xmlns=\"http://www.w3.org/1999/xhtml\">\n\t<p>General Overview / Document
Scope</p>\n</div>",
      "Title": "General Overview / Document Scope"
    },
    {
      "Attribute Type": "Heading",
      "Contributor": "kit7fe",
      "Created On": "2019-10-08T06:18:45.662Z",
      "Creator": "kit7fe",
      "Description": "",
      "Identifier": 629020,
      "Modified On": "2019-10-08T06:18:45.662Z",
      "ReqIF.Text": "<div
xmlns=\"http://www.w3.org/1999/xhtml\">\n\t<p>Document Scope</p>\n</div>",
      "Title": "Document Scope"
    },
    {
      "Attribute Type": "Information",
      "Contributor": "kit7fe",
      "Created On": "2019-10-08T06:18:45.662Z",
      "Creator": "kit7fe",
      "Description": "",
      "Identifier": 629016,
      "Modified On": "2019-10-08T06:18:45.662Z",
      "ReqIF.Text": "<div
xmlns=\"http://www.w3.org/1999/xhtml\">\n\t<p>&lt;put below a first
description of the scope for ECU requirement specification&gt;
</p>\n</div>",
      "Title": "<put below a first description of the scope for software
requirement specification>"
    },
    {
      "Attribute Type": "Heading",
      "Contributor": "kit7fe",
      "Created On": "2019-10-08T06:18:45.662Z",
      "Creator": "kit7fe",
      "Description": "",
      "Identifier": 629012,
      "Modified On": "2019-10-08T06:18:45.662Z",
      "ReqIF.Text": "<div
```

```
xmlns=\"http://www.w3.org/1999/xhtml\">\n\t<p>Document Specific
Glossary</p>\n</div>",
      "Title": "Document Specific Glossary"
    },
    {
      "Attribute Type": "Information",
      "Contributor": "kit7fe",
      "Created On": "2019-10-08T06:18:45.677Z",
      "Creator": "kit7fe",
      "Description": "",
      "Identifier": 629013,
      "Modified On": "2019-10-08T06:18:45.677Z",
      "ReqIF.Text": "<div
xmlns=\"http://www.w3.org/1999/xhtml\">\n\t<p>&lt;put below a definition of
first glossary specific terms&gt;</p>\n</div>",
      "Title": "<put below a definition of first glossary specific terms>"
    },
    {
      "Attribute Type": "Heading",
      "Contributor": "kit7fe",
      "Created On": "2019-10-08T06:18:45.662Z",
      "Creator": "kit7fe",
      "Description": "",
      "Identifier": 629019,
      "Modified On": "2019-10-08T06:18:45.662Z",
      "ReqIF.Text": "<div
xmlns=\"http://www.w3.org/1999/xhtml\">\n\t<p>System
Requirements</p>\n</div>",
      "Title": "System Requirement"
    },
    {
      "Attribute Type": "Information",
      "Contributor": "kit7fe",
      "Created On": "2019-10-08T06:18:45.677Z",
      "Creator": "kit7fe",
      "Description": "",
      "Identifier": 629017,
      "Modified On": "2019-10-08T06:18:45.677Z",
      "ReqIF.Text": "<div
xmlns=\"http://www.w3.org/1999/xhtml\">\n\t<p>\n\t\t<br></br>\n\t\t<br>
</br>\n&lt;infos relevant for the complete chapter&gt;Note to the
template:- the example requirements below are independent of each other and
are showing the different possibilities of the requirements
structure\t</p>\n</div>",
      "Title": "<infos relevant for the complete chapter>"
    },
    {
      "Attribute Type": "Heading",
      "Contributor": "kit7fe",
      "Created On": "2019-10-08T06:18:45.646Z",
      "Creator": "kit7fe",
      "Description": "",
      "Identifier": 629022,
      "Modified On": "2019-10-08T06:18:45.646Z",
```

```
      "ReqIF.Text": "<div xmlns=\"http://www.w3.org/1999/xhtml\">\n\t<p>ECU
1st Requirement</p>\n</div>",
      "Title": "ECU 1st Requirement"
    },
    {
      "Allocation": "SW Allocation",
      "Attribute Type": "MO_FUNC_REQ",
      "CRQ": "RQONE03587423",
      "Contributor": "MIG1COB",
      "Created On": "2019-10-08T06:18:45.662Z",
      "Creator": "kit7fe",
      "Description": "",
      "Identifier": 629015,
      "Modified On": "2023-05-23T02:41:23.510Z",
      "ReqIF.Text": "<div
xmlns=\"http://www.w3.org/1999/xhtml\">\n\t<p>&lt;description of the
requirement in requirements language&gt;
</p>\n\t<p>\n\t\t<b>VEHICLE_SYSTEM_BEHAVIOUR</b>\n\t\t<b>CONSTRAINT</b>\n\t
\t<b>IMPACT</b>\n\t\t<b>INFO</b>\n\t\t<b>ASSUMPTION</b>\n\t\t<br>
</br>\n\t\t<br></br>\n\t\t<br></br>\n\t\t<br></br>\n\t\t<br></br>\n\t\t<br></br>\n\t\t<br>
</br>\n\t\t<br></br>\n\t\t<br></br>\n\t\t<br></br>\n\t\t<br></br>\n\t\t<br></br>\n\t\t<br>
</br>\n\t\t<br></br>\n\t\t<br></br>\n\t\t<br></br>\n\t\t<br></br>\n&lt;Optional:
description of desired vehicle behaviour (\"development
target\")&gt;\u00a0\u00a0 \u00a0 \u00a0&lt;Optional: constraints on the
solution space for the requirement&gt;\u00a0 \u00a0&lt;Optional:
description of possible cross-functional impact of the requirement, or
impact on other components&gt;&lt;Optional: additional informations about
the requirement:- know-how- background- HW dependencies related to the
system requirement- internal signals- etc&gt;&lt;Optional: assumptions on
the requirement&gt;\t</p>\n\t<p></p>\n</div>",
      "Safety Classification": "ASIL A",
      "Status": "New/Changed",
      "Title": "<description of the requirement in requirements language>",
      "VAR_FUNC_SYS": "VAR_Func_sys value",
      "Verification Criteria": "Test Environment:\nTest Bench/Lab-car with
hardware setup\n\nSuccess Criteria: Verify whether the signal value is
correct or not"
    },
    {
      "Attribute Type": "Heading",
      "Contributor": "kit7fe",
      "Created On": "2019-10-08T06:18:45.677Z",
      "Creator": "kit7fe",
      "Description": "",
      "Identifier": 629018,
      "Modified On": "2019-10-08T06:18:45.677Z",
      "ReqIF.Text": "<div
xmlns=\"http://www.w3.org/1999/xhtml\">\n\t<p>System Non Functional
Requirements</p>\n</div>",
      "Title": "System Non Fonctional Requirements"
    },
    {
      "Allocation": "Non_Func Allocation",
      "Attribute Type": "MO_NON_FUNC_REQ",
```

```
      "CRQ": "RQONE03587423",
      "Contributor": "MIG1COB",
      "Created On": "2019-10-08T06:18:45.677Z",
      "Creator": "kit7fe",
      "Description": "",
      "Identifier": 629014,
      "Modified On": "2023-05-23T02:56:54.487Z",
      "ReqIF.Text": "<div
xmlns=\"http://www.w3.org/1999/xhtml\">\n\t<p>&lt;description of the non
functional requirement in requirements language&gt;</p>\n</div>",
      "Safety Classification": "ASIL B",
      "Status": "New/Changed",
      "Title": "<description of the non functional requirement in
requirements language>",
      "VAR_FUNC_SYS": "Var_func_sys value 2",
      "Verification Criteria": "Non Func Test Environment:\nTest Bench/Lab-
car with hardware setup\n\nSuccess Criteria: Verify whether the signal
value is correct or not"
    }
  ]
}
```


- And same as file `Json_Output_Sample.json` or output of the `Task1` in `Round 2`.