**Research paper on Automated evaluation of Subjective Answer Sheets**

## Implementation-

Data set- Typed answers and paragraphs
Summarizing tool- Text Summarizer | Quill Bot AI

**Ques- What is Iterative waterfall model.**
**Typed Answer paragraph-**
Iterative Waterfall Model has the same phases as in the Classical Waterfall Model. But in Iterative Waterfall Model, Feedbacks paths are provided from every phase to its previous phases as shown in figure. The feedback paths allow the correction of errors find during a phase which is detected in a later phases. However, in this model, there is no feedback path to the feasibility study phase i.e., errors in the feasibility study phase cannot be corrected latter.
For example: If during testing a design error is identified, then feedback path allows us to change and rework the design documents.

**Summarize text using QuillBot AI Tool (text-1)**
The Classical Waterfall Model has feedback routes that connect each step to its predecessors. The feedback pathways enable the rectification of faults discovered during one phase that are discovered during other phases. However, there is no feedback route to the feasibility study step, therefore mistakes made during the original design phase cannot be fixed afterwards.

- Searching the question from question paper and parsing the URL to get text. This is called Web scraping.

**Code in python using BeautifulSoup 4 to parse and extract text from URL-**

```
#Step-0 Installed- bs4, requests, html5lib
import requests
from bs4 import BeautifulSoup
url    =    https://www.geeksforgeeks.org/software-engineering-iterative-waterfall-model/

#Step-1 Get the HTML
r = requests.get(url) #r=content
htmlContent = r.content
#Step-2 Parse the HTML
```

```
soup = BeautifulSoup(htmlContent, 'html.parser')

#Step-3 HTML tree traversal
#Get all the paras from page
paras = soup.find_all('p')
print(paras)

#get the text from the tags/soup
print(soup.find('p').get_text()) #paras text
```

- Output of about code is below-

**Google text(answer) after parsing**

In a practical software development project, the classical waterfall model is hard to use. So, the Iterative waterfall model can be thought of as incorporating the necessary changes to the classical waterfall model to make it usable in practical software development projects. It is almost the same as the classical waterfall model except some changes are made to increase the efficiency of the software development. The iterative waterfall model provides feedback paths from every phase to its preceding phases, which is the main difference from the classical waterfall model. When errors are detected at some later phase, these feedback paths allow correcting errors committed by programmers during some phase. The feedback paths allow the phase to be reworked in which errors are committed and these changes are reflected in the later phases. But there is no feedback path to the stage – feasibility study, because once a project has been taken, does not give up the project easily. It is good to detect errors in the same phase in which they are committed. It reduces the effort and time required to correct the errors.

- Again, using summarizer tool, we will summarize the text obtained. Output is below.

**Summarize text (text-2)**

In a practical software development project, the classical waterfall model is hard to use. The iterative waterfall model provides feedback paths from every phase to its preceding phases. It reduces the effort and time required to correct errors made during an early stage of a project's development.

- Now finding the similarity between both the Summarize text obtained i.e., text-1 and text-2 using **Cosine Similarity Index**.

**Cosine similarity in textual data** is used to compare the similarity between two text documents or tokenized texts. So, in order to use cosine similarity in text data, the raw text data has to be tokenized at the initial stage, and from the tokenized text data a similarity matrix has to be generated which can be passed on to the cosine similarity metrics for evaluating the similarity between the text document.

**Code to calculate Cosine Similarity b/w text-1 and text-2 in python-**

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer()

Document1 = """The Classical Waterfall Model has feedback routes that connect each step to its predecessors. The feedback pathways enable the rectification of faults discovered during one phase that are discovered during other phases. However, there is no feedback route to the feasibility study step, therefore mistakes made during the original design phase cannot be fixed afterwards"""

Document2 = """In a practical software development project, the classical waterfall model is hard to use. The iterative waterfall model provides feedback paths from every phase to its preceding phases. It reduces the effort and time required to correct errors made during an early stage of a project's development"""

corpus = [Document1, Document2]
Counts = count_vect.fit_transform(corpus)
pd.DataFrame(Counts.toarray(),columns=count_vect.get_feature_names_out(),index=['Document1', 'Document2'])

from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()
trsfm=vectorizer.fit_transform(corpus)
```

```
pd.DataFrame(trsfm.toarray(),columns=vectorizer.get_feature_na
mes_out(),index=['Document 1','Document 2'])

from sklearn.metrics.pairwise import cosine_similarity
result = cosine_similarity(trsfm[0:1],trsfm)
print("Cosine Similarity: ", result)
```

**Output**

```
PS C:\Users\HP\Desktop\Research paper for project\code> python
Cosine Similarity:  [[1.        0.3469342]]
PS C:\Users\HP\Desktop\Research paper for project\code>
```

The above result shows that there is 34% similarity between two texts i.e., text-1 and text-2. This is how Cosine Similarity Index works.

Next is deciding scores to be given.
To get scores, we multiply cosine similarity by 10 to get number like 3.469….
Now we will make classes between 0-9-
0-3, 4-6, 7-9. Using this, we will get an approx. range of marks.