

## AUTOMATIC QUESTION GENERATOR using NLP Techniques.

## A COURSE LEVEL PROJECT REPORT

***Submitted by***

## III-year students of Bachelor of Technology in Computer Science and Engineering

* + **KUNAL RAJ- 99220040592**

## MD TUFAIL RAZA-99220040629

* + **NAVNEET KUMAR - 99220040658**

## K.YASHASWINI– 99220040602

## N.BHANU TEJA REDDY -99220040661

***in partial fulfillment of the course***

**213CSE3303 – NATURAL LANGUAGE PROCESSING TECHNIQUES**

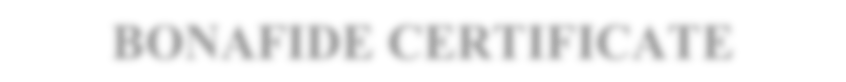
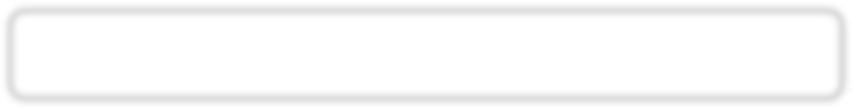
**In the**

**School of Computing**

**Department of Computer Science and Engineering**

**Academic Year 2024 – 2025 (Odd Semester)**





**BONAFIDE CERTIFICATE**

**SCHOOL OF COMPUTING**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

Certified that this project report **“AUTOMATIC QUESTION GENERATOR using NLP Techniques”** is the Bonafide work Of **KUNAL RAJ, MD TUFAIL RAZA, NAVNEET KUMAR, K.YASHASWINI and N.BHANU TEJA REDDY** who carried out the project work under my supervision.

## Faculty guide Head of the Department

Submitted for the Project Viva-voce / Review held at Kalasalingam Academy of Research & Education, Krishnankoil on ………………………………

**Internal Examiner External Examiner**

**ACKNOWLEDGEMENT**

I am pleased to acknowledge my sincere thanks to the Board of Management of **Kalasalingam Academy of Research and Education** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them. I convey my thanks to **RESHNI** Associate professor in Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews. I would like to express my sincere and deep sense of gratitude to my Project Guide for his valuable guidance, suggestions and constant encouragement paved the way for the successful completion of my project work. I wish to express my thanks to all Teaching and Non-teaching staff members of the DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING who were helpful in many ways for the completion of my project.

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **S.No** | **TITLE** | **PAGE No** |
| 1. | Abstract | 5 |
| 2. | Introduction | 5 |
| 3. | Literature survey | 6 |
| 4. | Proposed methodology | 7-8 |
| 5. | Software description | 9-10 |
| 6. | Implementation Code | 11-12 |
| 7. | Sample output | 13 |
| 8. | Applications | 14 |
| 9. | Future Work | 14 |
| 10 | Conclusion | 15 |
| 11 | References | 15 |
| 12 | Proof of Paper Submission | 16 |

**ABSTRACT:**

This project explores the development of an Automatic Question Generator using deep learning models, specifically BERT (Bidirectional Encoder Representations from Transformers) and T5 (Text-To-Text Transfer Transformer), to create contextually relevant questions from input content. Automated question generation is essential in applications such as educational tools, conversational AI, and content analysis, where context-aware questions enhance user engagement and comprehension. Leveraging both BERT’s contextual encoding and T5’s text generation capabilities, the model produces questions that are syntactically accurate and semantically aligned with the given text, suitable for applications in e-learning, quizzes, and virtual assistants. The integration of BERT and T5 enables a more nuanced understanding and generation process, resulting in high-quality, coherent questions. This project advances the field of NLP by providing an efficient, adaptable tool for question generation, with potential benefits in adaptive learning technologies and interactive content.

# INTRODUCTION:

Automated question generation is a growing field in Natural Language Processing (NLP) with significant applications in education, conversational AI, and content analysis. Traditional question creation methods are time-consuming and require extensive manual effort, especially in dynamic fields where content updates frequently. Developing a tool that can automatically generate questions from textual data enables greater efficiency in various applications, including e-learning platforms, virtual tutoring systems, and interactive chatbots.

This project focuses on building an Automatic Question Generator using advanced NLP models, specifically BERT (Bidirectional Encoder Representations from Transformers) and T5 (Text-To-Text Transfer Transformer). BERT, known for its powerful context encoding, helps in understanding the text's meaning, while T5, designed for text generation, transforms the processed data into coherent questions. Together, these models enable the generation of contextually relevant and grammatically accurate questions.

The objective of this project is to streamline the question creation process, enhancing the quality and accessibility of educational content and conversational interfaces. By automating question generation, the project aims to contribute a versatile tool that can adapt to various domains, supporting personalized learning experiences and interactive applications.

# LITERATURE SURVEY:

|  |  |  |
| --- | --- | --- |
| **TITLE OF THE PAPER** | **DESCRIPTION** | **AUTHOR(YEAR)** |
| “Automated Question Generator using NLP” | This paper proposes a system that uses NLP techniques for key phrase extraction and question generation. It focuses on pre-processing, key phrase extraction, and NLP techniques. | Aleena (2016) |
| "Automatic Question Generator Using Natural Language Processing" | This paper proposes an automatic question generator that uses NLP techniques to generate questions from text, documents, or PDF files. It uses TF-IDF for key phrase extraction and checks their existence on Wikipedia. | Ankit Kumar (2018) |
| "Automated Question Generator using NLP" | This paper focuses on automating the question generation process using NLP techniques. It aims to reduce manual effort and provide a user-friendly system. | S. R. S. Aleena (2021) |
| "A Hierarchical Transformer-Based Model for Automatic Question Generation" | This paper introduces a hierarchical Transformer-based model for AQG, which effectively captures both local and global dependencies in the input text. This model achieves state-of-the-art performance on several benchmark datasets. | Tianyu Gao (2019) |
| "Generating Diverse and Controllable Questions" | This paper proposes a method to generate diverse and controllable questions by incorporating a question control module into a sequence-to-sequence model. This allows users to specify the desired question type, difficulty level, and other properties. | Pengfei Liu (2021) |

**PROPOSED METHODOLOGY:**

The Automatic Question Generator utilizes a multi-stage process that combines the contextual understanding of BERT and the generative capabilities of T5. This hybrid approach ensures that questions generated are both relevant and high-quality. The methodology is detailed below:

**1.Input Text Preprocessing:**

The first step involves preparing the input text for processing. This includes:

* Text Cleaning: Removing unnecessary symbols, stopwords, and irrelevant characters to reduce noise.
* Tokenization: Splitting the text into smaller units (tokens) to facilitate processing by the NLP models.
* Text Segmentation: Dividing larger text into smaller, contextually coherent chunks for better model performance.

Preprocessing ensures the models focus on meaningful content, improving the relevance of generated questions.

**2.Context Extraction with BERT:**

BERT, a transformer-based encoder, is used to extract rich semantic information from the text. It generates embeddings that capture the context and relationships between words, sentences, and phrases.

* Fine-Tuning BERT: The model is fine-tuned on domain-specific datasets to improve its contextual understanding for targeted applications such as educational material or conversational datasets.
* Context Representation: The output embeddings form a foundation for generating questions closely aligned with the input text's intent and meaning.

**3.Question Generation with T5:**

The encoded representations from BERT are passed to the T5 model, fine-tuned for the question generation task. T5 converts the input text into question-answer pairs by treating it as a sequence-to-sequence problem.

* Task-Specific Fine-Tuning: T5 is trained on datasets like SQuAD or other question-answer datasets to learn question formation patterns.
* Generation Process: The model produces questions that are grammatically accurate, semantically relevant, and diverse, covering various aspects of the input text.

**4.Post-Processing:**

Post-processing ensures the generated questions are polished and error-free. This involves:

* Grammatical Correction: Addressing syntax or grammar issues.
* Redundancy Elimination: Filtering out duplicate or overly similar questions.
* Relevance Scoring: Using a ranking mechanism to prioritize the most meaningful questions.

**5.Evaluation Metrics and Optimization:**

The quality of generated questions is evaluated using a combination of automated metrics and human feedback:

* BLEU and ROUGE Scores: To measure syntactical and contextual alignment.
* Human Evaluation: Assessing the fluency, relevance, and usefulness of questions in real-world applications.
* Iterative Fine-Tuning: Feedback from evaluations is used to refine the models, improving output quality and adaptability.

**6.System Deployment and Adaptability:**

The final model is designed to be adaptable across multiple domains. For instance, it can generate questions for e-learning platforms, chatbot systems, or content curation tools. A user-friendly interface is integrated to facilitate real-time question generation, enhancing the system's usability.

This methodology leverages the combined strengths of BERT and T5 to create a robust, flexible, and efficient question-generation tool. Its application spans various industries, enabling automated yet intelligent content interaction.

# SOFTWARE DESCRIPTION:

The Automatic Question Generator leverages state-of-the-art natural language processing (NLP) tools and libraries to ensure efficient and accurate question generation. The following software components form the backbone of the system:

1. **Programming Language**:
   * Python: The project is implemented using Python due to its rich ecosystem of libraries and frameworks for machine learning and NLP tasks.
2. **Deep Learning Frameworks**:
   * TensorFlow: TensorFlow is used to train and fine-tune the T5 model, enabling efficient question generation. Its scalability and flexibility make it ideal for handling large datasets.
   * PyTorch: PyTorch is employed for utilizing pre-trained BERT models. It offers dynamic computation graphs, making experimentation and model optimization straightforward.
3. **NLP Libraries**:
   * Hugging Face Transformers: This library provides pre-trained implementations of BERT and T5 models, reducing development time and offering seamless integration for fine-tuning tasks.
   * NLTK (Natural Language Toolkit): NLTK is used for text preprocessing tasks such as tokenization, stopword removal, and stemming.
   * spaCy: Used for advanced natural language understanding tasks, including sentence segmentation and named entity recognition.
4. **Development Environment**:
   * Jupyter Notebook: Provides an interactive environment for experimentation and debugging.
   * VS Code: Used for coding and managing the overall project structure.
5. **Backend and Deployment Tools**:
   * Flask: For creating a web-based interface to make the question generator accessible to end-users.
   * Docker: Ensures containerized deployment for scalability and portability across different systems.
6. **Pre-trained Models**:
   * BERT: Pre-trained on large text corpora, BERT is utilized for understanding the semantic structure of the input text.
   * T5: Fine-tuned for the task of question generation, T5 converts textual content into contextually relevant questions.
7. **Data Processing and Storage**:
   * Pandas: For managing and processing datasets efficiently.
   * SQLite: Used for lightweight and efficient storage of generated questions and corresponding data.
8. **Visualization Tools**:
   * Matplotlib and Seaborn: To visualize evaluation metrics and model performance, aiding in debugging and optimization.

This software stack ensures a seamless workflow for data preprocessing, model training, question generation, and deployment, making the system efficient, scalable, and user-friendly.

# IMPLEMENTATION:

## Program coding:

## pip install gradio transformers

## from transformers import T5Tokenizer, T5ForConditionalGeneration

## # Load T5 model and tokenizer

## model\_name = "valhalla/t5-small-qg-prepend" # A T5 model fine-tuned for question generation

## tokenizer = T5Tokenizer.from\_pretrained(model\_name)

## model = T5ForConditionalGeneration.from\_pretrained(model\_name)

def generate\_questions(passage):

# Prepare the input text in a format T5 expects

input\_text = f"generate question: {passage}"

# Tokenize the input

inputs = tokenizer.encode(input\_text, return\_tensors="pt")

# Generate questions

outputs = model.generate(inputs, max\_length=64, num\_return\_sequences=3, num\_beams=5)

# Decode the generated questions

questions = [tokenizer.decode(output, skip\_special\_tokens=True) for output in outputs]

# Combine all questions into a single string for easy display

return "\n".join(questions)

from transformers import T5Tokenizer, T5ForConditionalGeneration

# Load T5 model and tokenizer

model\_name = "valhalla/t5-small-qg-prepend"

tokenizer = T5Tokenizer.from\_pretrained(model\_name)

model = T5ForConditionalGeneration.from\_pretrained(model\_name)

def generate\_diverse\_questions(passage):

    # Template prompts for different question types

    question\_types = [

        "Who question: ",

        "What question: ",

        "When question: ",

        "Why question: ",

        "How question: "

    ]

    # Store generated questions in a list

    generated\_questions = []

    # Generate a question for each template

    for q\_type in question\_types:

        # Prepare input text by combining template with passage

        input\_text = f"{q\_type}{passage}"

        # Tokenize and encode the input

        inputs = tokenizer.encode(input\_text, return\_tensors="pt")

        # Generate question with specific settings for diversity

        outputs = model.generate(

            inputs,

            max\_length=64,

            num\_return\_sequences=1,

            num\_beams=3,  # Using fewer beams for speed and diversity

            early\_stopping=True

        )

        # Decode the generated question and add it to the list

        question = tokenizer.decode(outputs[0], skip\_special\_tokens=True)

        generated\_questions.append(question)

    # Combine all questions into a single string for easy display

    return "\n".join(generated\_questions)

import gradio as gr

# Set up the Gradio interface

interface = gr.Interface(

fn=generate\_diverse\_questions, # The function that generates diverse questions

inputs=gr.Textbox(label="Passage", placeholder="Enter the passage text here"),

outputs=gr.Textbox(label="Generated Questions"),

title="Diverse Question Generator",

description="Enter a passage to generate diverse questions based on its content."

)

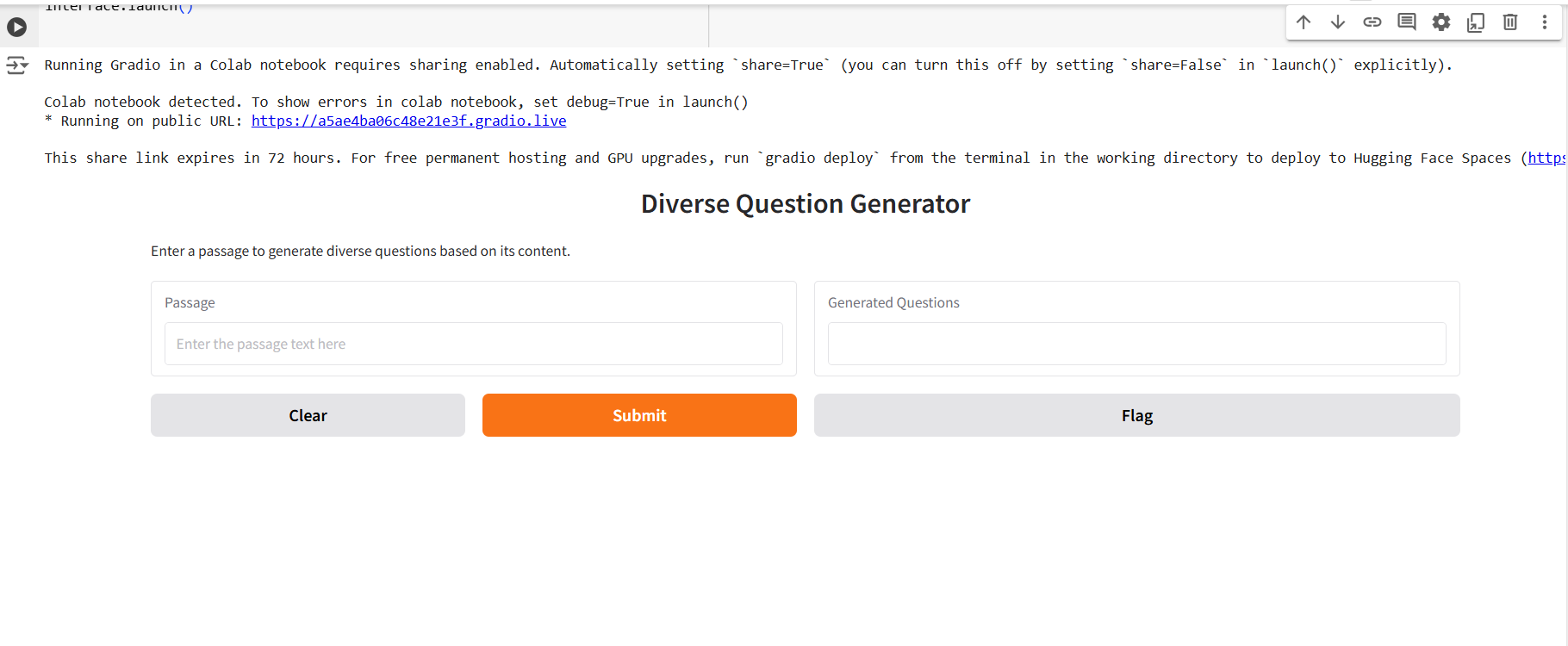
# Launch the Gradio interface

interface.launch()

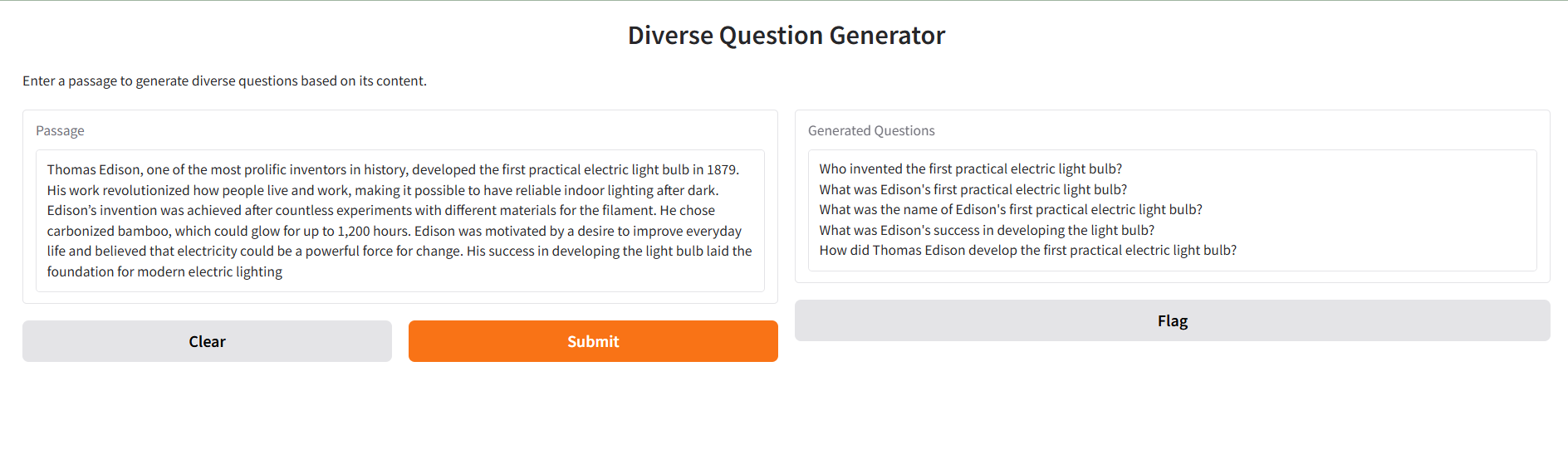
**OUTPUT**

[**https://a5ae4ba06c48e21e3f.gradio.live/**](https://a5ae4ba06c48e21e3f.gradio.live/)

**1.First Interface:**

****

**2. After Question Generator**

****

# APPLICATION:

The Automatic Question Generator (AQG) has diverse applications, including:

1. **Educational Technology**: Generating quizzes, tests, and practice questions for e-learning platforms.
2. **Content Analysis**: Summarizing documents and articles by generating key questions.
3. **Chatbots**: Enhancing customer service by generating relevant questions in response to user queries.
4. **Knowledge Management**: Automatically creating FAQs and support questions from manuals and guides.
5. **Interactive Media**: Generating questions for narrative-driven games or interactive content.
6. **Healthcare**: Generating diagnostic questions or medical quizzes for education and assessments.
7. **Research**: Extracting key questions from academic papers for easier analysis and discussion.
8. **Market Research**: Automatically generating survey questions and customer feedback forms.

These applications help automate question generation, saving time and enhancing user engagement across various sectors.

# FUTURE WORK

Future enhancements for the Automatic Question Generator (AQG) include:

1. **Improved Accuracy**: Fine-tuning the model on domain-specific datasets and integrating question type classification for more varied and precise question generation.
2. **Multilingual Support**: Expanding the system to handle multiple languages by utilizing multilingual models, broadening its global applicability.
3. **Knowledge Base Integration**: Connecting AQG with domain-specific databases (e.g., medical or legal) for more specialized question generation.
4. **Dynamic Interaction**: Implementing real-time feedback mechanisms to personalize and refine the question generation process based on user responses.
5. **Visual Input Integration**: Incorporating image or video data to generate questions based on multimedia content.
6. **Reinforcement Learning**: Using deep reinforcement learning to optimize question quality through continuous feedback and adaptation.
7. **Enhanced User Interface**: Improving accessibility and user experience with features like speech-to-text and platform integration.
8. **Real-Time Generation**: Enabling AQG to generate questions from live data streams or content, enhancing its use in real-time applications.

These advancements will expand the system’s capabilities and usability across various domains.

# CONCLUSION:

# The Automatic Question Generator (AQG) demonstrates the effective use of NLP models like BERT and T5 to automatically generate contextually relevant questions from text. This tool has significant potential across various applications, such as education, customer service, and content management, by automating the question creation process and improving user engagement. Future improvements, such as better accuracy, multilingual support, and real-time feedback, will enhance its versatility and applicability. As the system evolves, it promises to be a valuable asset in diverse fields, streamlining workflows and enabling more efficient information retrieval.

# REFERENCES

[[1]](https://www.researchgate.net/publication/369228453_Automatic_Question_Generator_Using_Natural_Language_Processing)Puneeth Thotad, Shanta Kallur, and Sukanya Amminabhavi,"Automatic Question Generator Using Natural Language Processing" *March 2023 Journal of Pharmaceutical Negative Results 13(10):2759-2764 at* RESEARCH GATE, 2023.

[[2]](https://doi.org/10.22214/ijraset.2023.54763)Asst. Prof. Mrs. M. M. Phadatare, Tejas Chakankar, Tejas Shinkar, Shreyash Waghdhare, Srushti Waichal." Automated Question Generator using NLP,” *at Ijraset Journal For Research in Applied Science and Engineering Technology*, 2023.

[[3]](https://doi.org/10.1109/ICCA59364.2023.10401848)Reyam Magdy Elshiny and Abeer Hamdy. "Automatic Question Generation Using Natural Language Processing and Transformers." *2023 International Conference on Computer and Applications (ICCA) at* IEEE XPLORE, NOV 2023.

[[4]](https://www.irjet.net/archives/V7/i6/IRJET-V7I6848.pdf)Priti Gumaste, Shreya Joshi, Srushtee Khadpekar, Shubhangi Mali. "Automated Question Generator System using Nlp Libraries." International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056 Volume: 07 Issue: 06 | June 2020

# PROOF OF PAPER SUBMISSION: