

Intel® Unnati Industrial Training Program

AI-Powered Quizzing App- ACADEMIND

Group Name: 404 Debuggable

Group Members:

Gayathri Girish

Agila Benedict

Akshay T Manoj

Mentored by: Dr. Jesna Mohan

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to **Intel Unnati** for providing me with the opportunity to participate in the **Industrial Training Program**, which has been an immensely enriching and insightful experience.

I extend my heartfelt thanks to **Dr. Jesna Mohan**, Associate Professor, Department of Computer Science and Engineering, [Your College Name], for their invaluable guidance and support throughout the course of the training. Under their mentorship, I worked on the project titled "**AI-Powered Interactive Learning Assistant for Classroom**", which helped me deepen my understanding of AI technologies and their practical applications in the education domain.

I am also grateful to my institution and the coordinators of the training program for facilitating this opportunity and ensuring a smooth learning process.

This training has significantly enhanced my technical skills, problem-solving abilities, and project development experience, and I look forward to applying these insights in future endeavors.

CONTENTS

Sno.	Title	Page
1	INTRODUCTION	
2	IMPLEMENTATION	
3	RESULT	
4	PROBLEMS FACED	
5	CONCLUSION	
6	FUTURE ENHANCEMENTS	
7	REFERENCES AND BIBLIOGRAPHY	

Problem Statement:-

AI-Powered Interactive Learning Assistant for Classroom

INTRODUCTION

In our current educational landscape, students face a monotonous cycle of assessments each academic year, aimed at evaluating their understanding and mastery of various subjects. This traditional model not only adopts a one-size-fits-all philosophy but also gives rise to significant challenges related to grading efficiency and educator workload. Teachers often dedicate countless hours to grading assignments, a time-intensive process that detracts from their ability to engage with students directly or focus on developing impactful instructional strategies.

To address these pressing issues, we propose the development of an innovative online testing platform designed to facilitate secure and convenient exam completion for students. By harnessing the power of cutting-edge artificial intelligence technology, this platform will offer instant grading capabilities for subjective questions. The project entails a comprehensive integration of detailed syllabi and curricular standards into the AI system, equipping it with the contextual understanding necessary for accurate assessment of student submissions.

Additionally, the platform will feature a user-friendly interface that enhances student interaction. This streamlined approach to assessment not only accelerates the grading process but also enriches the overall educational experience. By automating mundane tasks such as question creation, answer key preparation, and grading, the AI system will free educators

from these burdens, enabling them to channel more time and energy into personalized instruction and support.

Ultimately, our goal is to significantly reduce the time and resources expended in traditional grading processes, while simultaneously creating a more effective and engaging learning atmosphere for students. By evolving the assessment approach, we aim to better cater to the individual needs of learners and enhance the quality of education they receive. The AI-driven platform has the potential to revolutionize the way assessments are conducted, fostering a more dynamic and interactive educational experience for both teachers and students.

IMPLEMENTATION

final.py

```

final.py > ...
1  import pdfplumber
2  from optimum.intel.openvino import OVModelForSeq2SeqLM, OVModelForQuestionAnswering
3  from transformers import AutoTokenizer, AutoModelForSeq2SeqLM, pipeline
4  from flask import Flask, render_template, request
5  from sentence_transformers import SentenceTransformer, util
6  import random
7  import re
8
9
10 def paragraph_chunking(text):
11     sentences = re.split(r'(?<=[.!?]) +', text.strip())
12     return [s.strip() for s in sentences if s.strip()]
13 def extract_text_from_file(file):
14     finaltext= ""
15     pdf=pdfplumber.open(file)
16     for i,j in enumerate(pdf.pages):
17         text=j.extract_text()
18         finaltext+=text+" "
19     return finaltext
20 file="ksp102.pdf"
21 f=extract_text_from_file(file)
22
23
24
25 t1 = AutoTokenizer.from_pretrained( "iarfmoose/t5-base-question-generator")
26
27 m1 = OVModelForSeq2SeqLM.from_pretrained( "iarfmoose/t5-base-question-generator",export=True)
28 m1.save_pretrained("IUIT SUMMER")
29 qam = OVModelForQuestionAnswering.from_pretrained("distilbert-base-cased-distilled-squad")
30 qat = AutoTokenizer.from_pretrained("distilbert-base-cased-distilled-squad")
31

```

```

final.py > ...
63 @app.route('/')
64 def index():
65     return render_template('indexdemo.html')
66
67 @app.route('/quiz', methods=['POST'])
68 def quiz():
69     roll = request.form['roll']
70     subject = request.form['subject']
71     return render_template('questions.html', roll=roll, subject=subject, shorts=short_answers)
72
73 @app.route('/submit', methods=['POST'])
74 def submit():
75     roll = request.form.get('roll')
76     subject = request.form.get('subject')
77     score = 0
78
79     for i in random_questions:
80         user_answer2 = request.form.get(f'short{i}', "").strip()
81
82         qa = pipeline("question-answering", model=qam, tokenizer=qat)
83         try:
84             result = qa(question=i, context=f)
85             expected_answer = result["answer"]
86         except:
87             expected_answer = ""
88
89         modelanswer= sample_answer_model.encode(user_answer2, convert_to_tensor=True)
90         expectedanswer = sample_answer_model.encode(expected_answer, convert_to_tensor=True)
91         print(modelanswer, expectedanswer)
92         similarity= util.cos_sim(modelanswer, expectedanswer).item()
93         if similarity >= 0.3:
94             score += 1
95
96     total = len(short_answers)
97     return render_template('result (1).html', score=score, total=total, roll=roll, subject=subject)

```

```

final.py > ...
31
32 def generate_questions(text):
33     input_text = "generate questions: " + text.strip().replace("\n", " ")
34     input_ids = t1.encode(input_text, return_tensors="pt", truncation=True, max_length=1024)
35
36     outputs = m1.generate(input_ids=input_ids,max_length=128,min_length=30,num_beams=3,num_return_sequences=3,no_repeat_ngram_size=3,early_stopping=True)
37
38     questions = t1.batch_decode(outputs, skip_special_tokens=True)
39     unique_questions= list(set([q.strip() for q in questions if "?" in q]))
40     final_questions=[]
41     for i in unique_questions:
42         quest=i.split("?")
43         final_questions.append(quest[0])
44     return final_questions
45
46 a = paragraph_chunking(f)
47 l = [ a [i:i + 10] for i in range(0, len(a), 10) ]
48 questions=[]
49 for i in l:
50     s=""
51     for j in i:
52         s+= " "+j
53
54     m=generate_questions(s)
55     questions.append(m[0])
56 num_samples=min(10,len(questions))
57 random_questions = random.sample(questions, num_samples)
58 sample_answer_model= SentenceTransformer('all-MiniLM-L6-v2')
59
60 app = Flask(__name__)
61 short_answers=random_questions
62

```

```

final.py > ... C:\Users\giris\IUIT SUMMER\templates\result (1).html
97     return render_template('result (1).html', score=score, total=total, roll=roll, subject=subject)
98 if __name__ == '__main__':
99     try:
100         app.run(port=5000, debug=True)
101     except:
102         print("Flask failed to start")
103
104

```

Backend Structure of the AI-Powered Quiz App

Introduction

The backend of our AI-powered quiz application is thoughtfully developed using Flask, a lightweight and flexible web framework that acts as a vital conduit between the frontend and backend functionalities. This well-

organized architecture enables us to effectively manage user inputs, ensuring that responses are processed swiftly and accurately. As a result, we are able to deliver a smooth and engaging experience for users as they navigate through our interactive quizzes.

Text Extraction

To generate questions for our quizzes, we begin by employing the `pdfplumber` module, a powerful tool that allows us to meticulously extract text from a designated PDF document. For instance, we applied this technique to a chapter from a Class 11 English textbook, which provided us with a rich and comprehensive reference to facilitate the generation of insightful questions. This method not only streamlines the process but also ensures that the questions created are closely aligned with the material presented in the document.

Question Generation and Answering

Our application leverages advanced models optimized with OpenVINO technology, namely `OVModelForSeq2SeqLM` and `OVModelForQuestionAnswering`. These sophisticated models play a crucial role in dynamically generating both questions and answers, enhancing the interactive experience for users. Once the models produce outputs, these responses are meticulously compared against the user's submissions to assess accuracy and provide meaningful feedback on their correctness. This process ensures an engaging and informative experience tailored to each user's input.

Frontend Integration with Flask

We created a web framework with Flask that efficiently manages frontend interactions and processes form data. This framework is vital in capturing user inputs smoothly, significantly enhancing the user experience.

Random Question Selection

To enhance user engagement, we implemented a random selection process using the `random` and `re` modules. This functionality picks 10 questions randomly from a larger set of prepared questions, adding an element of unpredictability to the quiz experience.

Maintaining Question Uniqueness

To generate meaningful and unique questions, we focused on individual paragraphs of the reference text. This method allowed us to maintain question relevance and minimize redundancy. The model `iarfmoose/t5-base-question-generator` was employed to create questions, while `distilbert-base-cased-distilled-squad` was used to generate their corresponding answers.

Logical Question Generation

During the question generation process, we noticed some illogical questions with improper punctuation. To address this, we tracked the patterns of

these faulty questions and utilized a simple string manipulation technique to refine the outputs.

Text Segmentation

To expedite the question generation, the reference text was divided into strings of 10 sentences each. This segmentation strategy not only sped up the process but also ensured that the questions derived were contextually relevant.

User Answer Evaluation

Once the user provided their answers, we implemented a sentence embedding model, `all-MiniLM-L6-v2`, to assess the similarity between user responses and AI-generated answers. A threshold similarity score of 0.3 was established; if the score exceeded this value, the user's answer was marked as correct.

Application Routing

The Flask app includes various routes that manage the flow of the application. The login page requests the user's roll number and subject to lead them to the quiz interface. Here, users answer the questions presented to them. After submission, the app evaluates the answers and calculates the similarity score, displaying the results to the user.

Conclusion

Finally, the Flask application is initiated on port 5000, ready to serve users and provide them with a dynamic and interactive quiz experience. This systematic backend structure ensures a robust and user-friendly application that effectively integrates AI capabilities into the learning process.

index.html

```

indexdemo.html 1 x  questions.html  final.py  result (1).html
templates > indexdemo.html > html > head > style > .form
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Quiz login page</title>
5      <style>
6          /* Basic reset and background setup */
7          body {
8              margin: 0;
9              padding: 0;
10             min-height: 110vh;
11             font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
12             background: linear-gradient(to bottom, #e6f0ff, #cce0f5, #b3d1f0, #f7f9fc);
13         }
14
15         /* Main wrapper for the form */
16         .form {
17             margin: 80px auto;
18             padding: 30px 25px;
19             max-width: 550px;
20             border-radius: 50px;
21             box-sizing: border-box;
22             box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
23             background-color: #ffffff;
24         }
25
26         .title {
27             text-align: center;
28             margin-bottom: 25px;
29             color: #002366;
30             display: flex;
31             flex-direction: column;
32             align-items: center;
33             gap: 10px;
34         }
35     </style>

```

```

indexdemo.html 1 x  questions.html  final.py  result (1).html
templates > indexdemo.html > html > head > style > .form
2  <html>
3  <head>
5      <style>
35
36         .info {
37             display: block;
38             margin-bottom: 6px;
39             font-weight: bold;
40             color: #002366;
41         }
42
43         /* Inputs and dropdowns */
44         input[type="text"],
45         select {
46             width: 100%;
47             padding: 12px 14px;
48             margin-bottom: 15px;
49             border: 1px solid #ccc;
50             border-radius: 6px;
51             font-size: 15px;
52             box-sizing: border-box;
53         }
54
55         /* Submit button */
56         input[type="submit"] {
57             align: center;
58             width: 100%;
59             padding: 12px;
60             background-color: #007bff;
61             color: #ffffff;
62             border: none;
63             border-radius: 6px;
64             font-size: 16px;
65             cursor: pointer;
66             transition: background-color 0.3s ease;
67         }
68     </style>

```

```

indexdemo.html 1 x  questions.html  final.py  result (1).html
templates > indexdemo.html > html > head > style > .form
2  <html>
3  <head>
5  <style>
68
69  /* Hover effect */
70  input[type="submit"]:hover {
71      background-color: #0056b3;
72  }
73
74  /* Mobile responsive adjustments */
75  @media (max-width: 500px) {
76      .container {
77          margin: 40px 20px;
78          padding: 25px 20px;
79      }
80  }
81  </style>
82  </head>
83  <body>
84      <div class="form">
85          <div class="title">
86              <h2>Quiz Login</h2>
87          </div>
88          <form action="/quiz" method="post">
89              <info for="roll">Roll No:</label>
90              <input type="text" id="roll" name="roll" required>
91
92              <info for="subject">Subject:</label>
93              <select id="subject" name="subject" required>
94                  <option value="" disabled selected>Select a subject</option>
95                  <option value="English">English</option>
96                  <option value="Manufacturing">Manufacturing</option>
97              </select>
98
99              <input type="submit" value="Start Quiz">
100          </form>
101      </div>
102  </body>
103  </html>

```

Objective

This block of code is used to create a responsive, visually pleasing website login page. It is designed in such a way that it creates a positive user friendly environment to collect the basic details of the user before taking part in the quiz.

Explanation

<!DOCTYPE html>

Is used to declare that the content file is an HTML5 (Hyper Text Markup Language) document.

The tag **<html>** marks the beginning of the HTML document.

The tag **<body>** is where the entire requirements and functioning of the page is implemented.

```
<div class = "form">
```

divides the login box, the styling of this box is done in the form sector within the style tag. The styling involves the setting up of the basic structure of the box, the padding, width and curving of the corner of the box, sizing and shadowing the box to make it visually appealing, and setting the background colour to white (#ffffff).

```
<div class = "title">
```

divides the title in the login box, and separate styling of the title is done in the title sector of within the <style> tag. It involves aligning the text and the items, setting the gap between the items to 10 pixels, and setting the colour to #002366. Flexbox is used to centre the contents in the box and space it accordingly. Within the <h2> tag we have the title within the login box, which is "Quiz Login".

```
<form action = "/quiz" method = "post">
```

this part of the code is used to send the entered data to the /quiz endpoint using the post method. In this way the data will be transferred securely from one page to the other.

```
<input type = "text" id = "roll" name = "roll" required>
```

through this block of code we set the valid input to be entered in the roll number field and ensures that the field is filled before proceeding to the next page. <label for = "roll">Roll No:</label> is used to make the "roll" more accessible. This is done by linking it to the user input data.

For subject selection we have the dropdown option. It is set is such a way that if the user does not fill the field the quiz will not proceed. The two

options that has been given for subject selection are Manufacturing and English. Label tag is used to make subject more accessible by linking it with the user input data.

The submit button is filed with the value Start Quiz, which on being pressed will check if the fields have been entered and then proceeds with the question-answer section. The submit button is styled with hover effect to make it user interactive and is styled effectively for the visual effects.

Questions.html

```
questions.html X indexdemo.html 1 final.py result (1).html
templates > questions.html > ...
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Quiz</title>
5   <style>
6     body {
7       margin: 0;
8       padding: 0;
9       font-family: Arial, sans-serif;
10      background: linear-gradient(to bottom, #e6f0ff, #cce0f5, #b3d1f0, #f7f9fc);
11      min-height: 100vh;
12    }
13
14    .container {
15      max-width: 600px;
16      margin: 80px auto;
17      padding: 30px 25px;
18      background-color: white;
19      border-radius: 8px;
20      box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
21      box-sizing: border-box;
22    }
23
24    .title {
25      text-align: center;
26      margin-bottom: 25px;
27      color: #002366;
28    }
29
30    h2, h3, h4 {
31      margin: 10px 0;
32    }
33
34    .question-block {
35      margin-bottom: 20px;
36    }
37
```



```

questions.html X indexdemo.html 1 final.py result (1).html
templates > questions.html > ...
2 <html>
3 <head>
5 <style>
37
38 .question-block p {
39     font-weight: bold;
40     margin-bottom: 8px;
41 }
42
43 .form-check {
44     margin-left: 15px;
45     margin-bottom: 6px;
46 }
47
48 label {
49     font-weight: normal;
50 }
51
52 textarea {
53     width: 100%;
54     padding: 10px;
55     font-size: 14px;
56     border: 1px solid #ccc;
57     border-radius: 6px;
58     resize: vertical;
59     box-sizing: border-box;
60     margin-top: 5px;
61 }
62
63 input[type="radio"] {
64     margin-right: 6px;
65     accent-color: #007bff;
66 }
67

```

```

questions.html X indexdemo.html 1 final.py result (1).html
templates > questions.html > ...
2 <html>
3 <head>
5 <style>
68     input[type="submit"] {
69         width: 100%;
70         padding: 12px;
71         background-color: #007bff;
72         color: white;
73         border: none;
74         border-radius: 6px;
75         font-size: 16px;
76         cursor: pointer;
77         margin-top: 20px;
78         transition: background-color 0.3s ease;
79     }
80
81     input[type="submit"]:hover {
82         background-color: #0056b3;
83     }
84
85     @media (max-width: 500px) {
86         .container {
87             margin: 40px 20px;
88             padding: 25px 20px;
89         }
90     }
91 </style>
92 </head>
93 <body>
94     <div class="container">
95         <div class="title">
96             <h2>Quiz</h2>
97             <h3>Roll No: {{ roll }}</h3>
98             <h4>Subject: {{ subject }}</h4>
99         </div>
100
101         <form action="/submit" method="post">
102             <input type="hidden" name="roll" value="{{ roll }}">
103             <input type="hidden" name="subject" value="{{ subject }}">

```

```

questions.html x indexdemo.html 1 final.py result (1).html
templates > <> questions.html > ...
2 <html>
93 <body>
94 <div class="container">
101 <form action="/submit" method="post">
106
107 <h4>Short Answer Questions</h4>
108 {% for q in shorts %}
109 <div class="question-block">
110 <p>{{ loop.index }}. {{ q }}</p>
111 <textarea name="short{{ loop.index0 }}" required></textarea>
112 </div>
113 {% endfor %}
114
115 <input type="submit" value="Submit">
116 </form>
117 </div>
118 </body>
119 </html>
120

```

Objective

The aim of designing and implementing this code is to create a user interactive page for the question answer section. This page is the one succeeding the login page and preceding the result page.

Explanation

<!DOCTYPE html>

Is used to declare that the content file is an HTML5 (Hyper Text Markup Language) document.

The tag <html> marks the beginning of the HTML document.

The tag <body> is where the entire requirements and functioning of the page is implemented.

<div class = "container"> divides the question container, the styling of this box is done in the container sector within the style tag. The styling involves

the setting up of the basic structure of the box and spacing the contents in the box, the padding, width and curving of the corner of the box, sizing and shadowing the box to make it visually appealing, setting the background colour to white (#ffffff) and setting the space between the margin and the content.

`<div class = "title">` divides the title and contents in the question container box, and separate styling of the title is done in the title sector of within the style tag. It involves aligning the text in the centre, setting the colour to #002366 and spacing the bottom with the margin up to 25 pixels. We have the header content within the h2 tag, h3 and h4 which is Quiz, dynamic fields using templating variables for roll number and subject respectively.

`<form action="/submit" method="post">` this part of the code is used to transfer the entered data to the /submit endpoint using the post method. In this way the data will be transferred securely for the evaluation process. `<input type="hidden" name="roll" value="{{ roll }}">` and `<input type="hidden" name="subject" value="{{ subject }}">` is used to securely retain the information: roll number and subject selection for further backend processing.

H4 has the Short Answer Questions. Through the process of looping through the list of questions that has been generated. `<textarea name="short{{loop.index0}}" required></textarea>` each questions will be displayed in the corresponding areas inside the question-block sector that is styled.

The submit button is styled and made visually appealing. The setup for answering the questions is in such a way that the submit button will not proceed the answers for evaluation unless and until all the answer field has been filled. On filling the whole answer fields the submit button will send the data for further processing.

Result.html

```
result (1).html X questions.html indexdemo.html 1 final.py
templates > result (1).html > ...
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Quiz Result</title>
5 <style>
6   body {
7     margin: 0;
8     padding: 0;
9     font-family: Arial, sans-serif;
10    background: linear-gradient(to bottom, #e6f0ff, #cce0f5, #b3d1f0, #f7f9fc);
11    min-height: 100vh;
12  }
13
14  .container {
15    max-width: 500px;
16    margin: 80px auto;
17    background-color: white;
18    border-radius: 8px;
19    padding: 30px 25px;
20    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
21    box-sizing: border-box;
22    text-align: center;
23  }
24
25  .title {
26    color: #002366;
27    margin-bottom: 20px;
28  }
29
30  h2, h3 {
31    margin: 10px 0;
32  }
33
34  .score {
35    font-size: 32px;
36    font-weight: bold;
37    color: #28a745;
38    margin-top: 20px;
39  }
```

result (1).html X questions.html indexdemo.html 1 final.py

templates > result (1).html > ...

```

2 <html>
3 <head>
4 <style>
5
41 .back-btn {
42     margin-top: 30px;
43     display: inline-block;
44     padding: 12px 20px;
45     background-color: #007bff;
46     color: white;
47     text-decoration: none;
48     border: none;
49     border-radius: 6px;
50     font-size: 16px;
51     cursor: pointer;
52     transition: background-color 0.3s ease;
53 }
54
55 .back-btn:hover {
56     background-color: #0056b3;
57 }
58
59 @media (max-width: 500px) {
60     .container {
61         margin: 40px 20px;
62         padding: 25px 20px;
63     }
64
65     .score {
66         font-size: 28px;
67     }
68 }
69 </style>
70 </head>
71 <body>
72     <div class="container">
73         <div class="title">
74             <h2>Quiz Result</h2>
75             <h3>Roll No: {{ roll }}</h3>
76             <h3>Subject: {{ subject }}</h3>

```

result (1).html X questions.html indexdemo.html 1 final.py

ter C:\Users\giris\UIT SUMMER\templates\result (1).html

```

2 <html>
3
71 <body>
72     <div class="container">
73         <div class="title">
74             <h2>Quiz Result</h2>
75             <h3>Roll No: {{ roll }}</h3>
76             <h3>Subject: {{ subject }}</h3>
77         </div>
78
79         <div class="score">Your Score: {{ score }} / {{ total }}</div>
80
81         <a href="/" class="back-btn">Try Another Quiz</a>
82     </div>
83 </body>
84 </html>
85

```

Objective

The aim of designing and implementing this code is to create a user interactive page for displaying the score, student roll no, subject. This page is the one succeeding the login page and has a return option to try another quiz.

Explanation

`<!DOCTYPE html>`

Is used to declare that the content file is an HTML5 (Hyper Text Markup Language) document.

The tag `<html>` marks the beginning of the HTML document.

The tag `<body>` is where the entire requirements and functioning of the page is implemented.

`<div class ="container">` divides the result container, the styling of this box is done in the container sector within the style tag. The styling involves the setting up of the basic structure of the box and spacing the contents in the box, the padding, width and curving of the corner of the box, sizing and shadowing the box to make it visually appealing, setting the background colour to white (#ffffff), text alignment, setting the space between the margin to auto and spacing to 80 pixels.

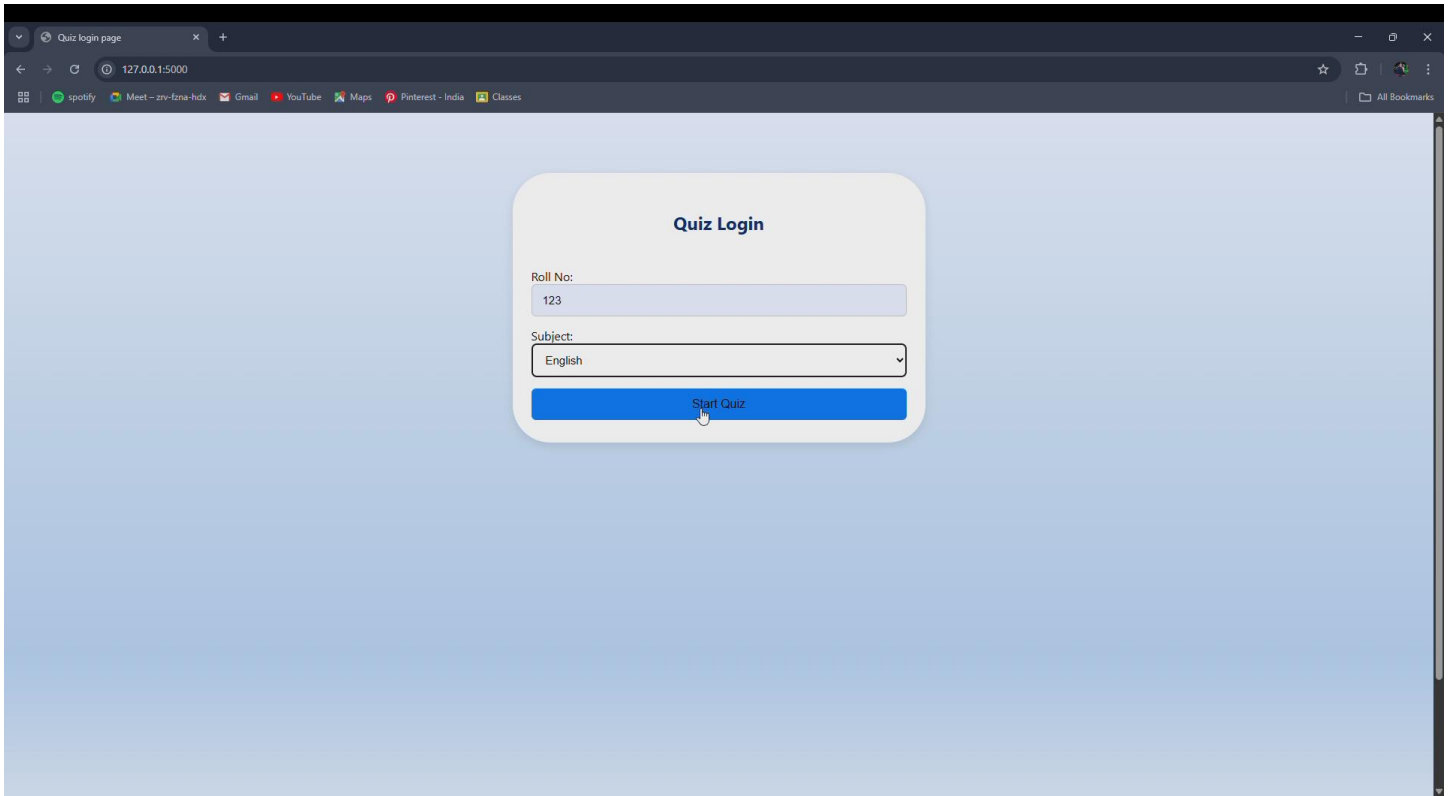
`<div class ="title">` divides the title and contents in the result container box, and separate styling of the title is done in the title sector of within the style tag. It involves setting the colour to #002366 and spacing the bottom with the margin up to 20 pixels. We have the header content within the h2 tag,

h3 and h4 which is Quiz Result, dynamic fields using templating variables for roll number and subject respectively.

`<div class="score">Your Score: {{score}}/{{total}}</div>` is used to display the actual score using `{{score}}` and the total marks for `{{total}}`. The styling of the result display is done within the score sector. It involves the font-size, boldness, colour of the text, and spacing from the margin.

`Try Another Quiz` is used to redirect the page to the initial page that is the login page. Through this action we will be able to try another quiz.

RESULTS



The screenshot shows a web browser window with the title 'Quiz login page'. The address bar displays '127.0.0.1:5000'. The browser's bookmark bar includes links to 'spotify', 'Meet - zny-fzna-hdx', 'Gmail', 'YouTube', 'Maps', 'Pinterest - India', and 'Classes'. The main content area features a 'Quiz Login' form with the following elements:

- Roll No:** A text input field containing the value '123'.
- Subject:** A dropdown menu with 'English' selected.
- Start Quiz:** A blue button with a mouse cursor hovering over it.

Quiz

Roll No: 123

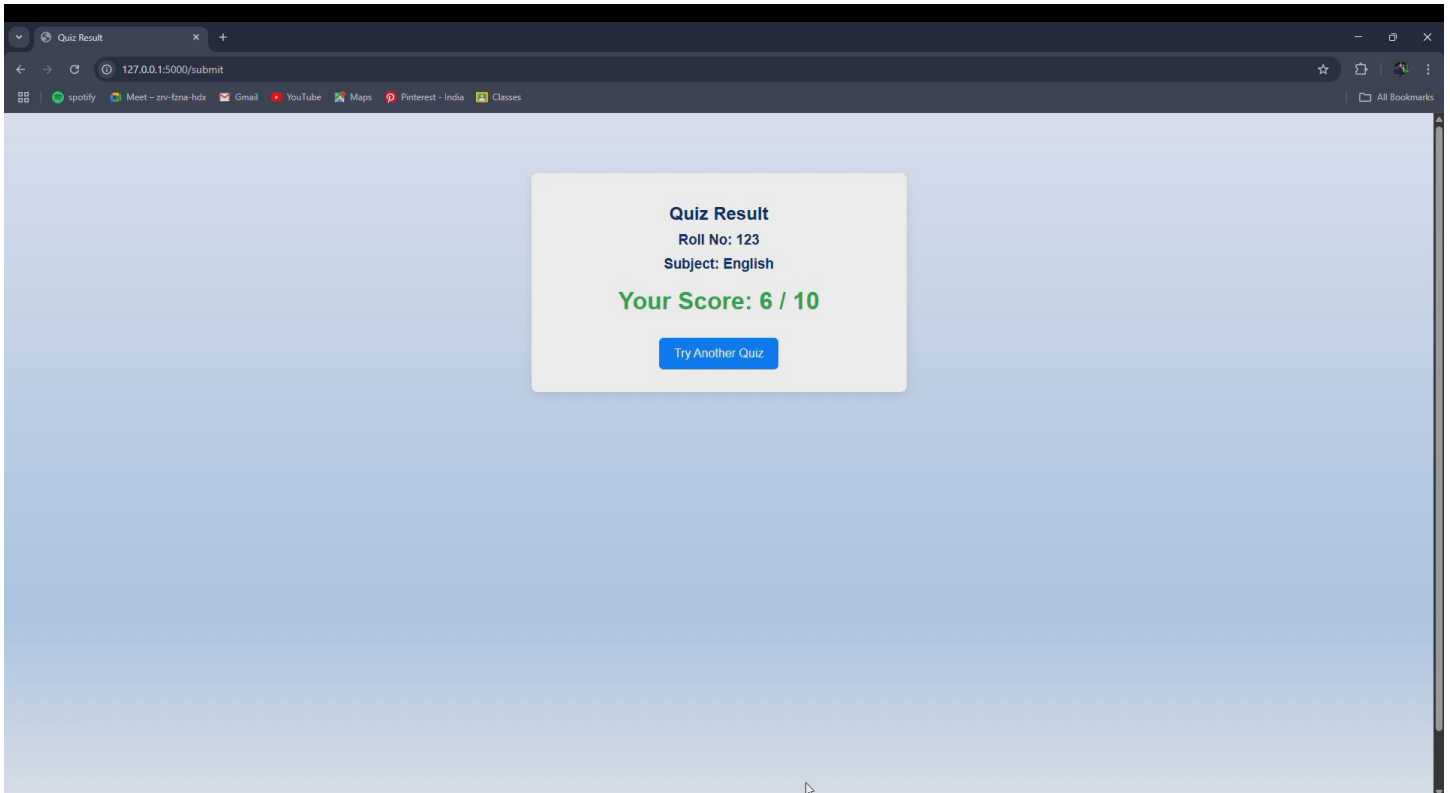
Subject: English

Short Answer Questions

1. What did the woman think of her mother
2. Who was Mrs Dorling
3. How did the woman remember the address
4. Why did the narrator of the story want to forget the address of the address in the story
5. What did the girl think of the apple on the pewter tablecloth
6. How did she decide to leave the station

the woman remembered the address just as her mother had mentioned it to her before she died.

4. Why did the narrator of the story want to forget the address of the address in the story
5. What did the girl think of the apple on the pewter tablecloth
6. How did she decide to leave the station
7. Why did the girl stop and stare at the table-cloth
8. Did she know that the cutlery we ate off every day was silver
9. What did she think of her daughter
10. Did she have a cup of tea for her



CHALLENGES FACED

1. One of the many challenges we faced was in the installation of OpenVINO. Among the members only one device was compatible with OpenVINO. We took some time to correctly install the OpenVINO.
2. We had an issue with text extraction from the pdf to generate questions. We were able to solve it by importing in the pdfplumber python library.
3. One of the issues we had to tackle was in the generation of questions. We had a hard time making it generate more than 3 questions and the generated questions were all similar with having only the grammar changed. To solve this we extracted sets of 10 lines from the attached pdf and generated a limited(like 2 or 3) questions from it. Then using the rand module we extracted 10 questions to print.
4. During the implementation of the frontend html files in the main code, we faced another setback. The questions were not getting displayed. We solved this by using a loop and printing the questions one by one.
5. The biggest hurdle we faced was in the grading system. First we used to compare the user input to the provided pdf, but this made it so that any answer including gibberish will give marks. We resolved it by using a question-answer bot to find the answer to the questions and then compare it with the user input, thus attaining a foolproof grading system.

CONCLUSION

This project is a successful demonstration of an AI-powered quiz system that automates both question generation and answer evaluation, thus reducing the effort required by the educators. Flask enables a user-friendly interface for the users and the backend ensures efficient processing of quiz data.

Overall, this project aims to offer a solution to the streamline assessment process by providing a scalable and educational value in real-world application by using cutting-edge AI tools.

FUTURE ENHANCEMENTS

1. Currently we have only implemented 2 subjects in our system. In the future, we aim to increase our subject-base.
2. Future version of the program will include data storage capabilities to retain quiz records of each student
3. We plan to upgrade our extraction system so that it can be able to extract the text from the images.
4. We plan to create a feedback system, which will help the student in recognising the areas he is weak in and also provide him with the knowledge of which all questions he answered incorrectly.
5. In the future, we plan to implement an option to let the student upload in the subject pdfs.

REFERENCES AND BIBLIOGRAPHY

1. [geeksforgeeks.org](https://www.geeksforgeeks.org)
2. chatgpt.com
3. huggingface.co
4. stackoverflow.com
5. intel.com