SOCAR Historical Document Processing Challenge

Hackathon Guidelines and Requirements

---

1. Challenge Background

1.1 Problem Statement

SOCAR possesses decades of valuable research data in the oil and gas domain. This historical data exists primarily in handwritten and printed formats across different languages and scripts. Currently, this knowledge is not easily accessible, searchable, or analyzable by different operational units.

1.2 Challenge Objective

Develop a complete solution that transforms historical handwritten and printed documents into an interactive, searchable knowledge base accessible through an intelligent chat agent interface for upstream use cases.

1.3 Business Value

- Enable discovery of historical research insights
- Make legacy knowledge accessible to current operations
- Support decision-making with historical context
- Preserve and digitize valuable institutional knowledge

---

2. System Requirements

2.1 Core Components

Your solution **must** include all three components:

1. **OCR (Optical Character Recognition) Processing Module**
   - Process PDF documents containing handwritten and/or printed text

    - Handle multiple alphabet
2. **Knowledge Base Creation**
    - Store processed information in a searchable format
    - Enable efficient retrieval of relevant information
3. **Intelligent Chat Agent**
    - Provide natural language answer for querying documents through endpoint
    - Return accurate, contextual responses based on document content
    - Include conversation history awareness

2.2 Input Data Specifications

Participants will work with PDF documents in **three difficulty categories**:

Document Type	Script/Language	Difficulty	Complexity Points
PDF Aze Print	Azerbaijani (Latin)	Easy	2 points
PDF Cyr Print	Azerbaijani (Cyrillic) / Russian	Medium	2 points
PDF Aze Hand	Azerbaijani (Handwritten)	Hard	6 points

Document Content: Historical research papers in oil and gas engineering domain

2.3 Technical Constraints

Infrastructure

- Virtual machines **without GPU** will be provided based on Azure VM
- Credentials will be shared with each team separately

External Resources

- Azure OpenAI API endpoint access will be provided
- API keys for multiple open-source LLM models available
- Teams may use these APIs for embeddings and language model inference

Deployment Preference

- **Open-source deployable solutions are strongly preferred (For example, LlaMa is open source while GPT is cloud)**
- Open-source deployable solutions receive **higher scores** than cloud-based solutions

3. Required Deliverables

3.1 Technical Solution

1. **Processing Pipeline**
   - Working OCR system (cloud or open-source)
   - Document preprocessing capabilities
   - Text extraction
2. **Chatbot**
   - Implemented vector database or equivalent
   - Efficient search and retrieval mechanism
   - LLM integration with knowledge system
   - Conversation management

3.2 Documentation (ReadME on GitHub)

- System architecture diagram
- Component documentation
- Reasonings of decision in component selection

3.3 Demonstration

- Explanation of the complete pipeline
- Show your reasonings on selected components
- Prepare maximum 5 minutes of presentation

---

4. Evaluation Criteria

4.1 OCR Benchmark (50% of total score)

Evaluation Method: OCR Benchmarking against ground truth

Metrics:

- **CER (Character Error Rate)**: Percentage of incorrect characters [0-1]
- **WER (Word Error Rate)**: Percentage of incorrect words [0-1]

Scoring (For document type refer to Section 2.2):

	Easy	Medium	Hard	Total
CSR	87,5	112,5	137,5	**337,5**
WSR	37,5	56,25	68,75	**162,5**
Total	**125**	**168,75**	**206,25**	**500**

Maximum points can be get from OCR benchmarks are

Benchmark Documents:

- 3 PDF benchmarks (one for each difficulty level) – not shared with teams
- Ground truth text will be used for benchmarking

4.2 Chatbot Benchmark (30% of total score)

Evaluation Method: LLM Judge evaluation

Test Format: Questions from each document type

- 2 Easy questions per document type
- 2 Medium questions per document type
- 1 Hard question per document type
- 1 Additional Question with chat history

Scoring Criteria (Evaluated by LLM Judge):

- **Answer**: Measures how accurately the final answer produced by the chatbot matches the Ground Truth. This score reflects whether the model correctly understood and responded to the question.
- **Citation Relevance**: Evaluates how relevant the retrieved chunks are to the Ground Truth. This score captures whether the retrieved information contains or could reasonably support reconstructing the correct answer.
- **Citation Order**: Assesses whether the retrieved chunks are arranged in a coherent, meaningful sequence that helps the model reconstruct the Ground Truth. Even if correct chunks are present, poor ordering can reduce answer quality.

Total Questions: 16 questions (5 questions × 3 document types + 1 question with chat history)

	Easy	Medium	Hard	Total
Printed	2	2	2	**6**
Cyrillic Printed	2	2	3	**7**
Handwriting	2	3	2	**7**
Total	**6**	**7**	**7**	**20**

Total Scores:

	Easy	Medium	Hard	Total
Printed	6,45	9,675	12,9	**58,05**
Cyrillic Printed	9,675	14,55	19,35	**106,5**
Handwriting	12,9	19,35	25,8	**135,45**
Total	**58,05**	**106,5**	**135,45**	**300**

4.3 Architecture and Implementation (20% of total score)

Deployment Approach:

- Open source deployable solution: Higher scores
- Cloud solution: Lower scores
- Hybrid approach: We are open to creative solutions

Technical Quality:

- Computational Load will be considered

Innovation:

- Novel approaches to OCR challenges
- Creative solutions for handwritten text
- Efficient resource utilization

---

5. API Specifications

Your solution must implement two REST API endpoints with the following specifications:

5.1 OCR Endpoint

Method

POST

Description

Accepts a PDF file upload and returns the extracted Markdown text for each page.

Input

A PDF file sent in ***content type*** `multipart/form-data`.

Response Format

A **list of dictionaries**, each representing a page of the PDF, with the following keys:

- **page_number** — *int*
  The page index starting from **1**, preserving the original PDF order.
- **MD_text** — *str*
  The Markdown-formatted extracted text for that page.

Example Response

```
[
  {
    "page_number": 1,
    "MD_text": "## Section Title\nExtracted markdown text..."
  },
  {
    "page_number": 2,
    "MD_text": "Additional extracted text..."
  }
]
```

5.2 LLM Endpoint

Method

POST

Description

Receives chat history and produces an LLM-generated answer along with source references.

Input

A JSON array representing chat history:

```
[
  {
    "role": "user",
    "content": "Message content"
  }
]
```

Response Format

A **dictionary** with two keys:

- **sources** — *list of dicts*
  Each item contains information about the referenced source text:
    - **pdf_name** — *str*
      Name of the PDF the text came from.
    - **page_number** — *int*
      The page number inside the PDF.
    - **content** — *str*
      The relevant extracted text (ideally in Markdown).
- **answer** — *str*
  The generated answer to the user query.

Example Response

```
{
  "sources": [
    {
      "pdf_name": "report.pdf",
      "page_number": 3,
      "content": "Extracted text snippet..."
    }
  ],
  "answer": "Here is the response based on the retrieved content..."
}
```

Good luck to all participants!

Making SOCAR's historical knowledge accessible for the future