PROJECT INTRODUCTION

SECTION 1: Previous Work and References

The objective of this project is to develop an application capable of generating and evaluating exam-style questions for the “Artificial Intelligence” course. The goal is to provide students with a practical tool for self-assessment and understanding of key AI problem-solving strategies.

The approach follows examples of **AI-assisted educational systems**, where large language models are used to generate problem sets and evaluate user responses. Similar methodologies can be found in the IBM report on *AI in Software Development* (IBM, 2023) and in research on **automatic question generation and grading** such as:

- Liu et al., *“Automatic Question Generation for Educational Purposes: A Survey”* (IEEE Access, 2020).
- Papamitsiou & Economides, *“Learning Analytics and Educational Data Mining in Practice”* (Computers in Human Behavior, 2014).
- OpenAI Documentation on “Function Calling” and “Text Analysis with LLMs” for content generation and evaluation.

These references show the general strategy of using **AI-assisted content creation** combined with **rule-based evaluation mechanisms**. Our system applies these principles to the specific domain of AI problems (search algorithms, game theory, etc.).

Automatic Question Generation (QG)

Language models such as **GPT**, **BERT**, and **T5** have been used to create questions from text.

Reference: Heilman, M., & Smith, N. A. (2010). *Good question! Statistical ranking for question generation.* NAACL HLT. https://aclanthology.org/N10-1086.pdf

Approach: Extract key facts from text, turn them into grammatically correct questions, and sort them by difficulty level.

Another related system: *Developing an Online Examination APP System* — IJET paper link

Automatic Answer Evaluation (Scoring or Grading)

These systems often use text similarity methods to compare a student’s answer with the correct one.

Reference: Mohler, M., Bunescu, R., & Mihalcea, R. (2011). *Learning to grade short answer questions using semantic similarity measures and dependency graphs.*

Approach: Combine keyword matching with meaning-based comparison.

Similar Applications

Examples of existing tools include **Socratic (by Google)**, **Quizlet**, and **EdStem QGen**, which use text processing techniques to generate and evaluate questions.

Official documentation and technical blog posts from these tools can also be cited to explain how they organize data and answers.

Frameworks and Libraries

Common technologies include **Python** and **Streamlit** for web interfaces, **PyPDF2/FPDF** for generating exam materials, and **JSON** for storing questions and answers. These are widely used in quick AI prototyping projects.

SECTION 2

Technical Logic and Algorithms for Question Generation and Evaluation System

Concept Extraction from Text

Goal: Identify key terms or concepts from the study material.

Algorithms: - **Word frequency (TF or TF-IDF):** Scan the text, count occurrences, exclude stopwords, and select the most frequent terms. (Greedy selection) - **Co-occurrence graph + PageRank:** Build a graph where nodes are words and edges represent co-occurrence; use a ranking algorithm to find the most central terms.

Question Generation

Goal: Generate grammatically correct questions based on extracted concepts.

Algorithms: - **Template-based generation:** Replace variables in templates such as “What is (concept)?” or “Define (concept)”.

Difficulty Classification

Goal: Assign a difficulty level (easy, medium, hard) to each question.

Algorithms: - **Rule-based heuristics:** Classify based on word count, type of question word, or structure. - **Decision tree:** Simple if-else logic modeled as a tree for clarity and modularity.

Answer Evaluation

Goal: Compare the user's answer with the correct one and assign a score.

Algorithms: - **Keyword matching:** Compare token sets; score based on overlap ratio. - **Edit distance (Levenshtein):** Compute minimal text edits required; implemented via dynamic programming. - **Fuzzy matching:** Tolerant string comparison using approximate search algorithms.

Exam Question Selection

Goal: Select a balanced subset of questions for each exam session.

Algorithms: - **Greedy selection:** Sequentially choose questions balancing difficulty and topic coverage. - **Backtracking:** Generate all valid combinations meeting given constraints (topics, number of easy/hard questions). - **Random sampling:** Simple selection for quick generation.

Report and PDF Generation

Goal: Create practice exams and performance reports in PDF format.

Algorithm: - **Sequential data traversal:** Loop through question objects and format them in the document. Linear time complexity (O(n)).

Summary of Algorithms Used

Module	Algorithms / Paradigms
Concept Extraction	TF-IDF / PageRank (Graph-based ranking)
Question Generation	Templates
Difficulty Classification	Rule-based / Decision Tree
Answer Evaluation	Set matching / Levenshtein distance
Question Selection	Greedy / Backtracking / Random
Report Generation	Sequential processing

This architecture ensures interpretability, efficiency, and ease of implementation using standard algorithmic techniques.