**EduGen: Creating Quizzes from Text-Based Content Using Python**

**Abstract**

EduGen was developed to help students prepare for exams and deepen their understanding of specific topics. By turning study materials into quizzes, the learning process becomes more interactive and engaging, rather than a repetitive task.

The application can also support teachers by automatically generating quiz questions from course materials and simplifying the grading process. Initially, the system will support files in formats like .docx or .pdf, with the goal of later expanding to handwritten or scanned notes.

The project is built using Python as the core language, with Django as the web framework, and Tailwind CSS for front-end styling.

1. **Introduction**

It is a common approach for students to practice for exams by pretending to be having it. For example, when I was a student, I would often prepare possible questions that might come across during the examinations, and write my answers for them. Another thing I would do was to ask other classmates to ask me random questions from the study materials. That was more engaging than simply re-reading the same information over and over again.

I believe that this tool could save a lot of time, and increase students’ motivation to learn.

Another advantage is that this can also be used by teachers. The program will be able to emphasize on specific parts of the material, as mentioned by the person who creates the quiz, during the design moment. They will also be able to choose what kind if questions they want: True/False, one answer, multi-answer.

The program accepts word documents or pdf files as input to be used for the quizz generation, but for the moment it can’t work with handwritten notes, or use AI-grading for open-ended answers.

**2. Theoretical Background**

* Basic concepts:
  + Natural Language Processing (NLP)
  + Question generation techniques
  + CRUD operations and database design
  + Text parsing/tokenization
* Related work (mention similar tools if needed)

**3. Requirements & Design**

* Functional requirements (e.g., upload text, generate questions, view quiz)
* Non-functional (e.g., fast generation, usable UI)
* Architecture diagram
* Database schema
* Technologies used (Flask/Django, SQLite/PostgreSQL, Python libraries)

**4. Implementation**

* Step-by-step how you built the system:
  + Frontend (if any)
  + Backend logic
  + Question generation algorithm
  + Storage and retrieval of quizzes
  + Optional: user accounts, difficulty levels

**5. Testing & Evaluation**

* Testing how well the questions are generated
* Sample use cases
* What worked well / challenges
* Limitations

**6. Conclusions and Future Work**

* What you achieved
* How it can be improved (e.g., better NLP, support for Romanian, more quiz types)