**SRS DOCUMENT**

**Introduction:**

**Objective:** This documentation provides the minimum functionality required of the “Dependency Aware Assessment Generator” called EduAssess.

**Project Scope:** The system will allow users to input a concept they wish to learn in Machine Learning. It will then fetch the prerequisites required to fully understand the concept and generate an assessment to test the users understanding of the prerequisite concepts.

**Environmental Characteristics:** The project utilizes LangGraph to dynamically generate context-aware questions, promoting adaptive learning with minimal environmental impact through efficient cloud-based processing.

**Overall Description:**

**Product Perspective:** EduAsses allows students to ensure they possess sufficient prerequisite knowledge before learning a new concept allowing them to systematically progress through their course work.

**Product Features:** The project aims to take a machine learning concept as input from the user and generate a list of questions which can be used to accurately understand the student’s foundation and assess whether they can proceed to that concept.

**User Classes:** The primary user of the concept is college students. Can also be used by professors and high school students interested in technology.

**Operating Environment:** The web-based system will run on standard web browsers and are recommended to be used only on laptops and desktops over hand-held devices.

**Design and Implementation Constraints:** The design needs to be scalable so that the increasing number of users and verification requests shall not degrade the performance of the product. It must also follow the tenets laid out in the DPDP act to protect student data.

**Functional Requirements:**

1. **Fetching prerequisites**

**Description:** The system accepts a machine learning concept provided as input through a dropdown and displays the prerequisite concepts.

**Input:** User selects the required concept

**Processing:** A list of prerequisites is fetched from a hardcoded concept graph.

**Output:** Prerequisites are displayed to the user.

1. **Generating Assessment**

**Description:** The system uses the prerequisites to create a quiz for the user to test their knowledge.

**Input:** The concept is selected from the dropdown list and a button labelled generate assessment is selected.

**Processing**: The prerequisites are fetched and sent to the backend LLM to generate the assessment.

**Output:** An assessment comprising of a mix of numerical, one word and multiple-choice questions is generated.

1. **Generating report**

**Description:** The user’s test results are analyzed to see if they can proceed to the new concept or if they need to review something, and if so what needs to be reviewed.

**Input:** The user’s answers to the generated quiz.

**Processing:** Use the user’s answers and compare it to the answer key in the backend to check for any wrong answers and generate a comprehensive report.

**Output:** The students assessment results and an analysis of whether they should proceed with the new topic is generated.

1. **Chatbot**

**Description:** Basic queries of the user can be answered using the chatbot.

**Input:** The user’s basic doubts.

**Processing:** The question is sent to an open source LLM using and API key which generates a response.

**Output:** Response is displayed to the user.

**External Interface Requirements:**

1. **API Interfaces**
   1. Interface to model to generate questions:
      1. Request:

{

"model": "meta-llama/Meta-Llama-3.1-8B-Instruct-fast",

"messages": [

{

"role": "user",

"content": "User-defined prompt text"

}

]

}

* + 1. Response

{

"choices": [

{

"message": {

"content": "Generated assessment text"

}

}

],

"other\_metadata": { /\* may include token usage, response time, etc. \*/ }

}

* 1. Interface to access chatbot:
     1. Not yet defined.

1. **Model Interfaces**
   1. Meta-Llama-3.1-8B-Instruct-fast is being used for generating questions with the agentic AI
   2. Mistral 7B is being used for the chatbot.

**Non-Functional Requirements:**

1. **Performance Requirements**
   1. **Response Time:** 
      * Fetch prerequisites in under 1 second
      * Generate assessment in under 20 seconds
      * Display assessment results in under 5 seconds.
   2. **Throughput:**
      * The project should be able to handle at least 3 simultaneous requests to the backend at a time.
2. **Reliability Requirements**
   1. **Error handling**

* Fallback mechanism for concepts that are not found.
* Backup methods to generate an assessment if the LLM does not work.

**2.2-Logging**

* Error logging with full stack traces
* Log the results of each call to backend LLM

**4.Hardware Requirements**

* Developers require a local system with at least 8GB RAM, 256GB storage and minimum intel i5 core.

**5.Scalability Requirements**

* Should be able to grow to support up to 10 users at a time.

**6.Quality Requirements**

* The generated assessment should have 80% relevance to the concept being tested, and it can be tested using an external LLM.

**7.Maintainibilty Requirements**

**7.1- Code Organization**

* Modular class structure (quiz\_generator, fetch\_prereqs)
* Clear separation of concerns between ML models and algorithmic backend

**8.Dependency Requirements**

* Hugging Face Inference API
* MERN stack components
* RAG for the chatbot depends on LLM