**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 27 June 2025 |
| Team ID | LTVIP2025TMID31153 |
| Project Name | Edu TutorAI-Personalized Learning with Generative AI and LMS Integration |
| Maximum Marks | 4 Marks |

**Technical Architecture:**

**Architecture Layers & Flow**

* User Interface Layer (Local)
* Technology: Gradio
* Purpose: Collects user inputs (concepts, language, topic, PDF)
* Screens: Login/Register, Classroom Interface, Concept/Quiz Display

**Application Logic Layer (Local)**

* Technology: Python
* Logic Handled:
  + Prompt generation
  + Session management
  + Quiz formatting
  + PDF parsing (PyPDF2)

**AI Model Layer (Cloud)**

* Technology: Hugging Face Transformers API
* Model Used: ibm-granite/granite-3.3-2b-instruct
* Function: Generates explanations, grammar lessons, quizzes
* Interface: Integrated using Hugging Face pipeline (text-generation)

**Data Storage Layer (Local)**

* Technology: Python in-memory dictionary (user\_sessions)
* Purpose: Stores user login state and session responses during runtime
* Future Scope: Extend to use Firebase or PostgreSQL for persistence

**External Interfaces**

* Hugging Face Model Access (via Token)
* PDF Reader: PyPDF2 parses uploaded files

**Optional Future Integrations: LMS APIs (Moodle, Google Classroom)**

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

**Table-1: Components & Technologies:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Description** | **Technology** |
|  | User Interface | Gradio web-based interface to interact with concept inputs, quiz generation, PDF upload | Gradio (Python-based UI) |
|  | Application Logic-1 | Main backend logic for processing inputs, managing sessions, formatting outputs | Python |
|  | Application Logic-2 | PDF reading and text extraction from uploaded documents | PyPDF2 (Python Library) |
|  | Application Logic-3 | Prompt construction and formatting for AI model | Python (Prompt Engineering) |
|  | Database | Temporary in-memory session tracking | Python Dictionary (Local memory) |
|  | File Storage | Uploaded PDFs stored temporarily during runtime | Local File System (via Gradio) |
|  | External API | API used to access a large language model | Hugging Face Transformers API (via Token) |
|  | Machine Learning Model | Used for generating content such as quizzes, explanations, grammar lessons | IBM Granite 3.3-2B-Instruct (via Hugging Face) |
|  | Infrastructure (Server/Cloud) | Application is executed in Jupyter/Colab or local environments | Google Colab / Local Python Runtime |

**Table-2: Application Characteristics:**

| **S.No** | **Characteristics** | **Description** | **Technology** |
| --- | --- | --- | --- |
|  | Open-Source Frameworks | Gradio (for UI), Transformers library for model interfacing | Gradio, Hugging Face Transformers |
|  | Security Implementations | Login-based access, username/password authentication, Hugging Face token for model access | Python Auth, HF Token |
|  | Scalable Architecture | Layered modular design (UI → Logic → Model), easily extensible with DB and LMS | Modular Python App, API-based Architecture |
|  | Availability | Hosted via Google Colab or local runtime; can be deployed on cloud for 24/7 availability | Google Colab / Flask + Hugging Face API |
|  | Performance | Optimized for fast responses (<3–5 seconds per query), uses Hugging Face’s inference optimization | Hugging Face Accelerated APIs, PyPDF2 |