**Requirement Analysis Phase**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 28 June 2025 |
| Team ID | LTVIP2025TMID36180 |
| Project Name | **Personalised learning with generative AI and Lms Integration** |
| Maximum Marks | 4 Marks |

**Technical stack:**

**1. Frontend Stack (User Interface)**

The frontend is responsible for the **user experience** of both students (citizens) and administrators.

**A. UI Framework/Library:**

* **Streamlit** (Primary choice):
  + **Why?**: Streamlit is a great choice for rapid prototyping and data-driven applications with real-time updates. It offers an easy-to-use framework for building interactive dashboards, making it ideal for the **Educator Dashboard** and **Student Portal**.
* **React.js / Vue.js** (Alternative for larger UIs):
  + **Why?**: For complex UI features like quizzes, reports, and interactive learning paths, **React.js** (or **Vue.js**) can provide better performance, flexibility, and scalability.

**B. Styling:**

* **Tailwind CSS**:
  + **Why?**: Tailwind CSS offers a utility-first approach to styling, providing flexibility and fast styling without the need to write custom CSS.
* **Material UI** or **Ant Design** (for React-based UIs):
  + **Why?**: Prebuilt UI components that speed up the development process.

**C. Charting and Visualizations:**

* **Plotly** or **Chart.js**:
  + **Why?**: Used to create interactive visualizations for dashboards (e.g., showing student performance, quiz results, KPIs).

**2. Backend Stack (APIs & Data Processing)**

The backend handles the logic, data management, and AI integrations.

**A. Framework:**

* **FastAPI**:
  + **Why?**: FastAPI is a modern, fast (high-performance) web framework for building APIs with Python. It’s great for asynchronous processing, which is critical when handling real-time data like quiz results, feedback, and dynamic content generation.
* **Python**:
  + **Why?**: Python is perfect for integrating with AI models, handling server-side logic, and processing large datasets efficiently.

**B. API Communication:**

* **GraphQL** (Optional):
  + **Why?**: Provides a flexible way to query and retrieve only the data required by the frontend, which can be useful when you need to fetch complex student or course data dynamically.
* **REST APIs**:
  + **Why?**: Standard and well-supported method for communication between the frontend and backend.

**C. Asynchronous Task Queue:**

* **Celery**:
  + **Why?**: For handling long-running tasks like **AI model processing**, **real-time quiz feedback generation**, and **recommendation engine updates** asynchronously.
* **Redis** (as message broker for Celery):
  + **Why?**: Used to manage real-time data and caching.

**D. User Authentication:**

* **OAuth 2.0** (via Google Login):
  + **Why?**: Secure and convenient method for users (students and admins) to log in using their Google credentials, which aligns with **Google Classroom integration**.
* **JWT Tokens**:
  + **Why?**: For session management and secure user authentication in APIs.

**3. AI & Machine Learning Stack**

AI and machine learning are critical to the **personalized learning** experience, dynamic quiz generation, and performance forecasting.

**A. AI Models:**

* **IBM Watsonx (Granite LLM)**:
  + **Why?**: The **Granite LLM** (from IBM Watsonx) provides **Natural Language Processing (NLP)** capabilities for generating personalized content like quizzes, feedback, and learning paths.
  + **Key Capabilities**: Text generation, personalized feedback, dynamic content creation, and automated question generation.
* **Custom ML Models** (Python-based):
  + **Why?**: For **forecasting**, **anomaly detection**, and **student performance prediction**.
  + **Libraries**:
    - **scikit-learn** for classic machine learning models (e.g., regression, clustering)
    - **TensorFlow** or **PyTorch** for deep learning models (e.g., performance forecasting, anomaly detection)
    - **XGBoost** for regression and classification tasks
    - **Prophet** (from Facebook) for **time series forecasting** (e.g., student performance over time)

**B. Recommendation Engine:**

* **Pinecone**:
  + **Why?**: **Pinecone** is a fully managed vector database for **semantic search**. It can be used to store **student embeddings** (representations of learning behaviors) and provide real-time recommendations for personalized learning paths.

**C. Training Infrastructure:**

* **GPU-enabled servers** (via cloud services like **AWS** EC2 instances or **Google Cloud AI Platform**):
  + **Why?**: For efficient **model training**, particularly for deep learning tasks.

**4. Data Storage & Database Stack**

The platform needs a reliable and scalable database for handling large amounts of user data, quiz scores, performance metrics, and policy documents.

**A. Relational Database:**

* **PostgreSQL**:
  + **Why?**: PostgreSQL is an open-source, powerful relational database system that can handle large datasets and complex queries. It's suitable for storing structured data like **user profiles**, **quiz scores**, and **course information**.
* **SQLAlchemy** (for Python ORM):
  + **Why?**: Provides an easy way to interact with the database via Python.

**B. NoSQL Database (for unstructured data, like learning embeddings):**

* **MongoDB**:
  + **Why?**: MongoDB is useful for storing large amounts of unstructured data, such as **user interaction logs**, **AI-generated feedback**, and **learning paths**.

**C. Vector Database:**

* **Pinecone**:
  + **Why?**: Used to store **learning embeddings** (vector representations of quizzes, student profiles, and learning content) for real-time **semantic search** and **recommendation engine**.

**D. Data Warehousing:**

* **Google BigQuery** or **AWS Redshift** (for analytics):
  + **Why?**: These cloud data warehouses are highly scalable and provide fast querying capabilities for large-scale **analytics** and **reporting**.

**E. Caching Layer:**

* **Redis**:
  + **Why?**: Used for caching commonly requested data (e.g., quizzes, student profiles) to improve response time.

**5. Cloud Infrastructure & DevOps**

Cloud services are essential for scalability, reliability, and high availability.

**A. Cloud Service Provider:**

* **AWS** / **Google Cloud** / **Microsoft Azure**:
  + **Why?**: These platforms provide all the services needed for infrastructure (e.g., compute, storage, networking) and offer **auto-scaling** and **high availability**.
  + Services:
    - **AWS EC2/GCP Compute Engine** (for app hosting)
    - **AWS RDS / Google Cloud SQL** (for PostgreSQL)
    - **AWS S3 / GCP Storage** (for file storage)
    - **AWS Lambda / Google Cloud Functions** (for serverless functions)

**B. Containerization & Orchestration:**

* **Docker**:
  + **Why?**: Containers ensure consistent deployment across environments (local, staging, production).
* **Kubernetes** (for orchestration):
  + **Why?**: Kubernetes automates deployment, scaling, and management of containerized applications, ensuring **reliability** and **scalability**.

**C. CI/CD:**

* **GitHub Actions** or **GitLab CI**:
  + **Why?**: For **Continuous Integration** and **Continuous Deployment**, automating tests, builds, and deployments.

**D. Monitoring & Logging:**

* **Prometheus** + **Grafana**:
  + **Why?**: For real-time monitoring and dashboards to track system health, performance, and resource utilization.
* **Sentry** (for error tracking):
  + **Why?**: To track errors and performance bottlenecks in real-time.

**6. Security & Compliance**

**A. Encryption & Secure Connections:**

* **SSL/TLS** for securing communication.
* **AES-256 encryption** for sensitive data at rest (e.g., student data, feedback).

**B. Regulatory Compliance:**

* **GDPR**, **FERPA** (for student data privacy).
* Regular **data audits** and **security assessments**.

**Summary of the Technology Stack :**

| **Component** | **Technology** |
| --- | --- |
| **Frontend** | Streamlit, React.js, Tailwind CSS |
| **Backend** | FastAPI, Python, Celery, Redis |
| **AI/ML Models** | IBM Granite LLM, TensorFlow, scikit-learn |
| **Database** | PostgreSQL, MongoDB, Pinecone |
| **Cloud** | AWS/GCP/Azure, Kubernetes |
| **CI/CD & DevOps** | Docker, GitHub Actions, Prometheus |
| **Security** | OAuth 2.0, JWT, SSL/TLS |