Project Design Phase-II

Technology Stack (Architecture & Stack)

Date: 26 June 2025
Team ID: [To be filled by user]
Project Name: Edu Tutor AI
Maximum Marks: 4 Marks

Technical Architecture:

The Edu Tutor AI system will be built as a cloud-native, microservices-based application designed to provide personalized learning experiences. The architecture emphasizes modularity, scalability, and integration with various AI and data services.

Architectural Description (Conceptual Diagram):

1. **User Layer:**
    - **User Interface (UI):** Users (students, educators) interact with the system primarily through a responsive **Web Application** and potentially **Mobile Applications (iOS/Android)**.
    - **Interaction:** Users submit queries, receive personalized content, view progress, and engage with the AI tutor.
2. **Edge Layer (Front-end Services):**
    - **Content Delivery Network (CDN):** Serves static assets (HTML, CSS, JavaScript, images) for fast global delivery.
    - **Load Balancer:** Distributes incoming user requests across multiple instances of the front-end and API Gateway services.
3. **Application Layer (Backend Microservices - Cloud):**
    - **API Gateway:** Acts as a single entry point for all client requests, routing them to the appropriate microservice. Handles authentication, rate limiting, and request validation.
    - **User Management Service:** Manages user profiles, authentication (e.g., OAuth 2.0), authorization, and session management.
    - **Learning Content Service:** Stores and manages educational content (text, videos, quizzes, interactive lessons). This service can also integrate with external content providers.
    - **AI Tutoring Service (Core Logic):**
        - **Natural Language Processing (NLP) Module:** Processes user queries, understands intent, and extracts key information using advanced NLP models.
        - **Personalization Engine (ML Model):** Analyzes student performance,

learning style, and progress to recommend personalized content and learning paths.
        - **Content Generation Module (ML Model):** Generates explanations, practice questions, and summaries based on retrieved content and student needs. (Leverages Large Language Models - LLMs).
        - **Dialogue Management:** Manages the conversational flow with the student.
    - **Assessment & Progress Tracking Service:** Records student responses, scores assessments, tracks learning progress, and generates performance reports.
    - **Recommendation Service:** Provides content recommendations based on collaborative filtering or content-based filtering.
    - **Notification Service:** Handles notifications to users (e.g., progress updates, new content availability).
4. **Data Layer (Cloud):**
    - **Primary Database (NoSQL/Document DB):** Stores user profiles, learning progress, content metadata, and interaction logs. Chosen for flexibility and scalability.
    - **Content Storage (Object Storage):** Stores large educational assets like videos, audio files, and large documents.
    - **Analytics Database/Data Warehouse:** Collects aggregated data for performance analytics, user behavior analysis, and model training.
    - **Vector Database:** Potentially used for efficient retrieval of semantically similar content for the AI tutoring service (e.g., for RAG - Retrieval Augmented Generation).
5. **External Interfaces (Third-party APIs):**
    - **Authentication Provider API:** For secure user authentication (e.g., Google, Microsoft SSO).
    - **Payment Gateway API:** If premium content or features are offered.
    - **Educational Content APIs:** Integration with external educational platforms or content libraries (e.g., Khan Academy API).
    - **Speech-to-Text/Text-to-Speech APIs:** For voice interactions within the AI tutor (if applicable).
6. **Machine Learning Operations (MLOps) Infrastructure (Cloud):**
    - **ML Model Training & Deployment:** Environment for training, versioning, and deploying AI/ML models (Personalization, Content Generation, NLP).
    - **Monitoring & Logging:** Tools for observing system health, performance, and AI model inference.

Infrastructural Demarcation (Cloud):

The entire application will be deployed on a cloud platform (e.g., Google Cloud Platform, AWS, Azure) utilizing managed services for databases, compute, storage, and AI/ML. This ensures high availability, scalability, and reduced operational overhead.

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1	User Interface	How users interact with the application (students, educators).	Web UI: React.js / Next.js (for SSR/SSG), HTML5, CSS3 (Tailwind CSS for styling) Mobile App (Optional): React Native (for iOS/Android)
2	Application Logic-1	API Gateway & User Management Logic.	Node.js with Express.js / Python with FastAPI (for microservices), JWT for authentication.
3	Application Logic-2	Core AI Tutoring Logic (NLP, Personalization, Dialogue Management).	Python (Primary Language for AI/ML), TensorFlow / PyTorch (ML Frameworks), Hugging Face Transformers (for NLP models), LangChain / LlamaIndex (for LLM orchestration/RAG).
4	Application Logic-3	Content Generation & Recommendation Logic.	Python (leveraging LLMs like Gemini Pro or custom fine-tuned models), Scikit-learn (for classical ML recommendation algorithms).
5	Database	Stores user data, content metadata, performance metrics.	MongoDB / Firestore (NoSQL Document Database) - for flexible schema and

			scalability.
6	Cloud Database	Managed database service on the chosen cloud platform.	Google Cloud Firestore (if GCP) / AWS DynamoDB (if AWS) / Azure Cosmos DB (if Azure).
7	File Storage	Stores large educational assets (videos, audio, large documents).	Google Cloud Storage (if GCP) / AWS S3 (if AWS) / Azure Blob Storage (if Azure).
8	External API-1	Authentication and Authorization.	Google Identity Platform / Firebase Authentication, Auth0, Okta.
9	External API-2	Integration with third-party educational content providers or rich media APIs.	Khan Academy API, YouTube API (for embedding videos), relevant academic APIs.
10	Machine Learning Model	Specific ML models used within the AI Tutoring and Personalization services.	NLP Models: BERT, GPT (for natural language understanding and generation). Recommendation Models: Collaborative Filtering (e.g., ALS), Content-Based Filtering. Personalization Models: Reinforcement Learning, Contextual Bandits.
11	Infrastructure (Server/Cloud)	Application Deployment on Cloud.	**Cloud Server Configuration:** Kubernetes (for container orchestration of microservices), Cloud

			Functions/AWS Lambda (for serverless components), Docker (for containerization). **Cloud Providers:** Google Cloud Platform (GCP) / Amazon Web Services (AWS) / Microsoft Azure.

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology Used
1	Open-Source Frameworks	List the open-source frameworks used.	React.js, Node.js/Python FastAPI, TensorFlow/PyTorch, Hugging Face Transformers, Kubernetes, Docker.
2	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	Authentication: OAuth 2.0, JWT (JSON Web Tokens). Authorization: Role-Based Access Control (RBAC). Data Encryption: HTTPS/TLS for data in transit, AES-256 for data at rest. Network Security: Cloud-native firewalls, Virtual Private Clouds (VPCs). OWASP Top 10 considerations in development. IAM Controls: Granular Identity and Access Management

			on cloud platform.
3	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services).	Microservices: Decoupled services allow independent scaling of individual components. Containerization (Docker) & Orchestration (Kubernetes): Enables elastic scaling of application instances based on demand. Cloud Databases (e.g., Firestore/DynamoDB) : Designed for horizontal scaling. Load Balancers: Distribute traffic efficiently.
4	Availability	Justify the availability of application (e.g., use of load balancers, distributed servers etc.).	Redundant Deployments: Microservices deployed across multiple availability zones/regions. Load Balancers: Distribute traffic to healthy instances. Managed Cloud Services: Databases, storage, and compute services managed by cloud providers with built-in high availability. Automatic Healing: Kubernetes automatically replaces failed containers/nodes.
5	Performance	Design consideration	CDN: For fast delivery

		for the performance of the application (number of requests per sec, use of Cache, use of CDN's) etc.	of static assets globally. Caching: In-memory caching (e.g., Redis) for frequently accessed data and API responses. Asynchronous Processing: Message queues (e.g., Kafka, RabbitMQ) for background tasks like content processing or ML model inference. Database Indexing: Optimized queries. Efficient Algorithms: For AI/ML models.

References:

- https://c4model.com/
- https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/
- https://www.ibm.com/cloud/architecture
- https://aws.amazon.com/architecture
- https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d