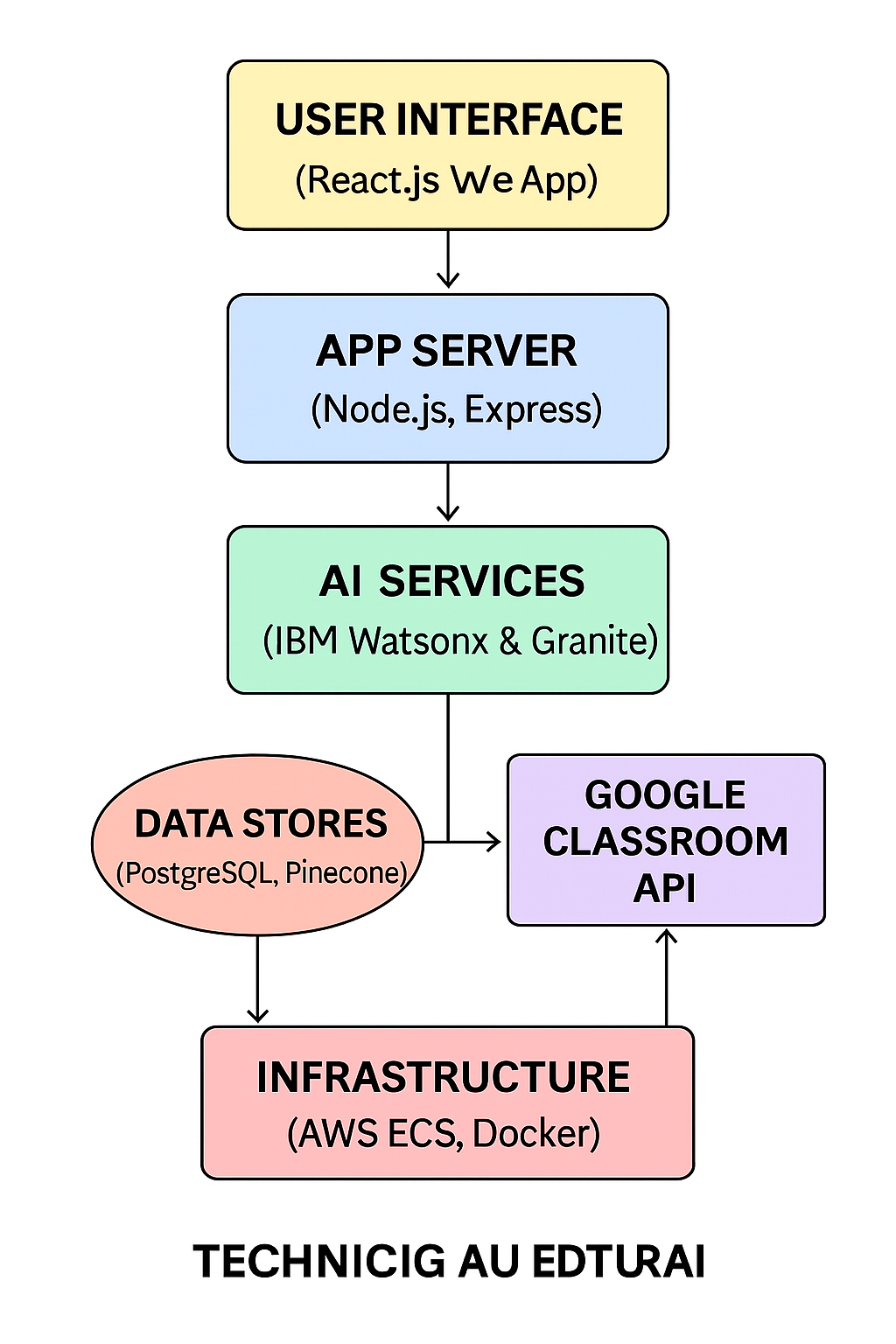
**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 26 June 2025 |
| Team ID | LTVIP2025TMID20451 |
| Project Name | EduTutor AI: Personalized Learning with Generative AI and LMS Integration |
| Maximum Marks | 4 Marks |

**Technical Architecture:**

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 | How the user interacts with the system | React.js, HTML5, CSS3, JavaScript, Redux |
|  | Application Logic-1 | Generates dynamic quizzes & feedback | Node.js, Express, Granite LLM, IBM Watsonx |
|  | Application Logic-2 | Calibrates difficulty & diagnostic scoring | Python, IBM Watsonx models |
|  | Application Logic-3 | Syncs roster and assignments via LMS | Node.js, Google Classroom API (OAuth2) |
|  | Database | Relational metadata storage | PostgreSQL |
|  | Cloud Database | Vector embeddings & analytics data | Pinecone Vector DB |
|  | File Storage | Stores logs, reports, and exports | AWS S3 |
|  | External API-1 | Fetches courses, students, and assignments | Google Classroom REST API |
|  | External API-2 | Sends confirmation, reminders, and reports | SendGrid (or AWS SES) |
|  | Machine Learning Model | LLM-based question generation and diagnostic assessment | IBM Granite foundation models; Watsonx NLU/NLG |
|  | Infrastructure (Server / Cloud) | Hosts and scales services | AWS ECS (Fargate), Docker, GitHub Actions CI/CD |

**Table-2: Application Characteristics:**

| **S.No** | **Characteristics** | **Description** | **Technology** |
| --- | --- | --- | --- |
|  | Open-Source Frameworks | Core frameworks used | Technology of Opensource framework |
|  | Security Implementations | Authentication, encryption, and access control | OAuth2, JWT, TLS 1.2+, AES-256, IAM roles |
|  | Scalable Architecture | Supports increasing load via modular microservices | AWS ECS auto-scaling, Docker containers |
|  | Availability | Ensures uptime and disaster recovery | Multi-AZ deployment, ALB load balancers, backups |
|  | Performance | Low latency and high throughput | Redis cache for sessions/results, AWS CloudFront CDN |

**References:**

[**https://c4model.com/**](https://c4model.com/)

[**https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/**](https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/)

[**https://www.ibm.com/cloud/architecture**](https://www.ibm.com/cloud/architecture)

[**https://aws.amazon.com/architecture**](https://aws.amazon.com/architecture)

[**https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d**](https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d)