**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 31 January 2025 |
| Team ID | SWTID1741256970149949 |
| Project Name | Rhythmic Tunes |
| Maximum Marks | 4 Marks |

**Technical Architecture:**

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

**Example:** Rhythmic Tunes

Below is an example of Table-1 outlining the technology stack for a music streaming app—expressed with a rhythmic tunes metaphor to illustrate how each component contributes to a harmonious performance.

**Table 1: Components & Technologies for a Music Streaming App**

| **Component** | **Technologies** | **Rhythmic Analogy** |
| --- | --- | --- |
| **User Interface (UI)** | React.js, Angular, Vue.js; React Native, Flutter (for mobile) | The lead melody that captivates users and sets the stage for the performance. |
| **Backend / API Server** | Node.js with Express, Python (Django/Flask), Java Spring Boot | The conductor orchestrating data flow and coordinating every part of the musical ensemble. |
| **Streaming Server** | Nginx RTMP, Wowza, AWS Elemental Media | The rhythm section ensuring smooth, uninterrupted playback—keeping the beat steady. |
| **Media Processing & Transcoding** | FFMPEG, AWS Elastic Transcoder, Azure Media Services | The studio where raw tracks are mixed and mastered to achieve optimal audio quality. |
| **Database** | PostgreSQL, MySQL, MongoDB | The library that archives every note and beat, making sure every detail is stored for future recall. |
| **Cache** | Redis, Memcached | The rapid percussion that provides immediate access to frequently used data, maintaining the pace. |
| **Message Broker / Real-Time Data** | Apache Kafka, RabbitMQ | The drummer delivering a steady pulse of real-time messages that keeps the performance in sync. |
| **Search Engine** | Elasticsearch | The DJ who quickly retrieves the perfect track from the vast archive, curating an ideal listening experience. |
| **Cloud Infrastructure** | AWS, Google Cloud, Microsoft Azure | The concert hall’s architecture, ensuring scalability and resilience as the audience grows. |
| **Security & Authentication** | OAuth 2.0, JWT, HTTPS | The backstage security, safeguarding every performance and ensuring only authorized access to the show. |
| **Monitoring & Logging** | Prometheus, Grafana, ELK Stack (Elasticsearch, Logstash, Kibana) | The sound engineer fine-tuning the performance by monitoring every note and resolving issues in real time. |
| **Content Delivery Network (CDN)** | AWS CloudFront, Cloudflare, Akamai | The amplification system that ensures every beat and note reaches the audience with crystal clarity. |
| **DevOps & CI/CD** | Docker, Kubernetes, Jenkins, GitLab CI/CD | The rehearsal and refining process that continuously polishes the performance, ensuring every update is seamless. |

**Table-2: Application Characteristics:**

Below is an example of Table-2, outlining key application characteristics for a music streaming app—presented with the metaphor of rhythmic tunes to illustrate how each aspect contributes to a seamless and harmonious performance.

**Table 2: Application Characteristics**

| **Characteristic** | **Description** | **Rhythmic Analogy** |
| --- | --- | --- |
| **Performance** | High throughput and low latency streaming ensuring smooth, continuous audio playback. | Like a tight drumbeat that keeps the rhythm steady and unbroken throughout the performance. |
| **Scalability** | Ability to scale horizontally and vertically to accommodate growing user numbers and concurrent streams. | Similar to an expanding orchestra that adds new instruments while preserving a unified sound. |
| **Availability** | High uptime (e.g., 99.9%) with 24/7 access across different regions, ensuring the service is always on stage. | Like a concert that never stops, ready to perform at any hour for its audience. |
| **Security** | Robust encryption, secure authentication (OAuth, JWT), and data protection measures to safeguard user data and streaming content. | Comparable to a fortified backstage area, protecting the precious musical scores and performers from intruders. |
| **Maintainability** | Modular architecture with clear documentation for smooth updates, debugging, and enhancements. | Like regularly tuning instruments to ensure every component of the performance stays in perfect harmony. |
| **Interoperability** | Seamless integration with third-party services, external APIs, and social media platforms for sharing and discovery. | As if guest artists join the ensemble, collaborating effortlessly to enhance the overall musical experience. |
| **Resilience** | Ability to handle errors gracefully, recover quickly from failures, and maintain data consistency during disruptions. | Much like a live musician’s improvisation—quickly adapting to unexpected changes while keeping the tempo intact. |
| **User Experience** | Intuitive interface design, quick navigation, personalized recommendations, and responsive performance across devices. | Like an engaging melody that resonates immediately with the audience, making every interaction feel natural. |
| **Real-Time Processing** | Efficient handling of live data for real-time recommendations, analytics, and social interactions. | Similar to a DJ adjusting the beat live, ensuring the energy and flow match the mood of the crowd. |
| **Data Integrity & Consistency** | Reliable data management to ensure accurate playback history, user preferences, and consistent streaming quality across sessions and devices. | Just as every note in a musical score is carefully placed, ensuring the symphony plays flawlessly from start to finish. |