Project Overview

You will develop an advanced Pastebin service for anonymous users. The service allows users to:

- **Create a Paste:** Enter a block of text to receive a randomly generated link.
- **Set Expiration:** Pastes are persistent by default, but users can optionally specify a timed expiration.
- **Retrieve a Paste:** View the contents of a paste using its unique URL.
- **Analytics & Monitoring:** The service tracks page views and aggregates monthly visit statistics.
- **Automated Cleanup:** Expired pastes are automatically deleted.
- **High Availability & Performance:** The system must be designed for scalability, low latency, and efficient resource usage.

Two Implementations

1. Monolithic Implementation

- ***Objective*:** Build a single-tier application that incorporates all functionality (paste creation, retrieval, expiration, analytics, cleanup, and caching) in one integrated solution.
- ***Requirements*:**
  - Develop a traditional monolithic application (without splitting into services or using external communication protocols like SOAP, REST, or gRPC for internal interactions).
  - Use a relational database (e.g., MySQL) pre-populated with data (if applicable) for storing paste contents, metadata (expiration, creation timestamp, etc.), and analytics data.
  - Provide a simple web interface and/or API endpoints for paste creation and retrieval.
- ***Evaluation*:**
  This version serves as your baseline to assess system performance, integration, and ease of development.

2. Microservices Implementation

- ***Objective*:** Re-architect the Pastebin service as a set of independent, distributed services. In this architecture, each service communicates with others using a specific technology. Your design decisions must be justified in your documentation.

- ***Integration & Deployment Requirements*:**
  - All microservices must be deployed independently (simulate a distributed environment using containers, VirtualBox/VMware, or a cloud platform).
  - Services should interact via their defined APIs and share common data sources where appropriate (e.g., MySQL for persistent data).
  - The system must be designed for high availability and scalability.

Performance & Comparative Testing

- **Test Scenario:** You must design and execute performance tests to simulate different loads on your system. Use a load testing tool (e.g., Locust or JMeter) to generate a range of concurrent requests.
- **Metrics to Measure:**
    - **Latency:** Average and peak response times.
    - **Throughput:** Number of requests processed per second.
    - **Resource Utilization:** Monitor CPU and RAM usage across different services.
    - **Scalability:** Ability to handle increased loads without degradation.
- **Evaluation:**
  The groups will be ranked based on the performance of their microservices system compared to the monolithic baseline. Key factors include:
    - Lower response times and higher throughput.
    - Efficient resource usage.
    - Impact of your design decisions and technology selections.

Submision

1. **Source Code**
2. **Design Document:** 3-5 pages detailing your architecture, technology choices, and design justification.
3. **Performance Test Scripts & Results:**
    - Load testing configurations, results in the form of charts/graphs, and a summary of performance metrics.
4. **Final Report & Presentation:**
    - A comprehensive report (5-10 pages) discussing your implementation, performance evaluation, comparative analysis, and recommendations.
    - A final presentation summarizing your project (slides + optional demo video).

Ranking

- **Competitive Aspect:** Your microservices implementation will be benchmarked under identical load testing conditions. The performance metrics, combined with design and documentation quality, will determine your group's ranking.
- **Ranking Factors:**
    - Lowest average and peak response times.
    - Highest throughput and efficient resource utilization.
    - Quality of design decisions and justifications (via final report + presentation).
- **Final Grade:** Your final grade will reflect both your technical implementation, and the measurable performance of your system compared to your peers.