CALIFORNIA STATE UNIVERSITY
FULLERTON

CPSC 449: Web Backend Engineering (Spring 2024)

Scalable Backend System with Secure Authentication

Overview

This project is exclusively focused on the backend aspect of web application development. You will build upon your previous assignment, enhancing it by integrating secure authentication, encrypted password storage, and the implementation of advanced scalable strategies to ensure the application can efficiently handle increased load and maintain high performance. Students are encouraged to explore various microframeworks and databases (SQL or NoSQL), tailoring their project to fit the chosen technologies.

Project Requirements

Secure User Authentication and Encrypted Password Storage:

1. Develop a secure user authentication mechanism. Consider methods like traditional username/password, token-based authentication (JWT), or OAuth2.
2. Encrypt passwords using encryption techniques to ensure the security of user data. (e.g. bcrypt)

Implement at Least Two Scalable Strategies:

1. **Caching** (e.g., Redis, Memcached): Implement caching to store frequently accessed data in memory, reducing database load and improving response times.
2. **Load Balancing** (e.g., Nginx, HAProxy): Use load balancing to distribute incoming traffic evenly across multiple servers, improving the responsiveness and reliability of your application.
3. **Microservices Architecture**: Break down the application into smaller, independently deployable services, each running its own

process and communicating through lightweight mechanisms. This approach allows for easier scaling and maintenance of individual components.

4. **Asynchronous Processing** (e.g., Celery, RQ): Offload time-consuming tasks to background workers or task queues, improving application throughput and user experience.
5. **Database Replication** (e.g. Litefs, MongoDB Replica Set): Use database replication to distribute read queries across multiple copies of the database, optimizing for read-heavy workloads.
6. **Database Sharding**: Partition your database into smaller, more manageable segments to distribute data across multiple servers, enhancing read/write performance.
7. **Horizontal Scaling** (e.g., Docker, Kubernetes): Add resources or server instances dynamically to accommodate increased load, using containerization technologies for efficient deployment and management.
8. **Auto-scaling**:
   Configure auto-scaling policies to automatically add or remove server instances based on predefined metrics (e.g., CPU usage, request rate).
   Use cloud provider services (e.g., AWS Auto Scaling, Google Cloud Autoscaler) to dynamically adjust resources based on demand.
9. **Event and Stream Processing**: Implement event-driven architecture and stream processing to handle real-time data processing efficiently. (e.g. Apache Kafka, Flink, Storm)
10. **Distributed Caching for Session Storage**: Use distributed caching mechanisms to manage user sessions, ensuring fast access to session data across multiple servers and maintaining session continuity even in high-traffic scenarios. (Redis, Memcached)
11. **Advanced Search Capabilities**: Incorporate a powerful search engine to offer quick, sophisticated search functionalities for large volumes of data. (e.g. Elasticsearch, Logstash)

Additional Information

- **This is a backend project only**: Focus on the server-side logic, data management, and scalable infrastructure without worrying about the frontend aspects.

- **Feel free to explore beyond the suggested strategies and technologies**: Innovative approaches to scalability and security are highly encouraged.

Bonus Points

- **Microservice Architecture and Asynchronous Messaging**: Refactor your application to a microservices architecture, utilizing asynchronous messaging for communication between services.

- **Database Migration**: Migrate your application's database from MySQL to MongoDB, carefully modeling the data to fit a NoSQL environment.

Team Collaboration

You may work in teams of up to 3 students. Divide project responsibilities among team members. Each member should be responsible for specific parts of the project, such as database management, API development, API testing, etc. Utilize version control (e.g., Git) for collaboration and to track project progress. Submit the name of your team members by the end of the week (04/14/2024).

Submission Guidelines

Deadline: 05/13/2024

Format: Submit your project via a **GitHub** link. Ensure your repository is well-organized, with a comprehensive README detailing setup instruction, scalable strategies implemented, and any significant design decisions. Moreover, submit a Zip file via Canvas ensuring that it includes **source code** and **Presentation** file and additional instruction for setup and running your application.
Team Submissions: Include a section in your **README** listing your team members and briefly describing each member's contributions.

Note: This project is tailored to elevate your backend development proficiency, with a concentration on scalability and security to equip you for practical, real-world software development challenges.