

Related Roadmaps

- ✓ Backend Roadmap
- ✓ System Design Roadmap

Backend Performance

Find the detailed version of this checklist
With details on how to implement these

<https://roadmap.sh>

Caching

- Utilize caching mechanisms (HTTP, Server/Client, CDN)
- Use cache aside, write-through, or read-through caching patterns based on your application requirements.
- Implement caching at various levels such as database query results, HTML fragments, or even full-page caching.
- Use proper cache-invalidation strategies to ensure data consistency and prevent stale content issues.

Databases

- Use connection pooling to reduce connection overhead
- Fine-tune connection pool settings (e.g. max connections idle timeout, connection reuse params) to optimize resource utilization and prevent connection exhaustion.
- Create efficient database indexes
- Keep an eye on and fine-tune ORM queries.
- Utilize features such as lazy loading, eager loading, and batch processing to optimize data retrieval.
- Implement efficient pagination for large datasets
- Avoid SELECT * queries and fetch only required columns
- Consider denormalizing database schema for ready-heavy workloads and reduce JOIN operations.
- Optimize JOIN operations and avoid unnecessary joins
- Regularly clean up unused data and perform database maintenance tasks like vacuuming, indexing, and optimizing queries.
- Enable slow-query logging and keep an eye on that.
- Set up database replication for redundancy and improved read performance.
- Use DB sharding for data distribution if required.
- Use profiling tools offered by your database.

Security

- Keep your dependencies up to date
- Implement proper authentication and authorization to prevent unauthorized access.
- Implement request throttling and rate limiting
- Regularly audit and update security measures

Performance Testing

- Conduct regular performance testing and benchmarking to identify performance regressions, track improvements, and fine-tune optimization efforts over time.

Optimize API Response

- Enforce reasonable payload size limits
- Enable compression for responses
- Implement efficient pagination for large datasets
- Minimise unnecessary processing or expensive computation on the server.

Asynchronism

- Offload heavy tasks to background jobs or queues
- Utilize message brokers for async communication between services.

Load Balancing & Scaling

- Use Horizontal or Vertical scaling whatever appropriate
- Use load balancing to distribute traffic across servers

Code Optimization

- Implement streaming of large requests/responses
- Profile your code to identify performance bottlenecks
- Optimize algorithms and data structures used
- Identify and optimize critical paths or frequently accessed endpoints for overall system health.
- Consider using compiled languages like Go or Rust for performance-critical parts of your backend.
- Look into different architectural styles (SOA, Micro services) and decompose services if required.
- Set appropriate connection timeouts and implement efficient retry mechanism to handle network issues
- Batch similar requests together to minimize overhead and reduce the number of round trips.

Monitoring and Logging

- Implement comprehensive monitoring and logging to track performance metrics and troubleshoot issues
- Use tools like Prometheus, Grafana, ELK stack.
- Use asynchronous logging mechanisms to minimise the logging overhead.

Network

- Minimize network latency by hosting your backend close to your users.
- Utilize HTTP keep-alive to reduce connection overhead
- Use CDNs for static and frequently accessed assets
- Prefetch or preload resources, data, or dependencies required for subsequent requests to minimise latency

Continue Learning with following relevant tracks

Backend Roadmap

System Design Roadmap