Task 2: Discussion task on system design

Scalability and parallelism

To ensure scalability and parallelism, a microservice architecture is used. Each component of the system is responsible for a separate aspect of functionality:

- **The game server** distributes the load across multiple nodes to support many game sessions simultaneously.
- **The caching system** provides quick access to frequently used data, reducing the load on databases.
- **Gateway API** manages request routing and provides load balancing between services.

Data consistency

The following approaches are used to ensure data consistency:

- **Database clustering** for load balancing and high availability.
- **Consensus protocols** (e.g., Paxos or Raft) for statefulness between replicas.
- **Data replication** to reduce delays in accessing data from different regions.

Real-time communication

For real-time communication, you can use:

- **WebSockets** for two-way communication between clients and servers.
- **Message queues** for handling traffic spikes and asynchronous request processing.

Security

- **Authentication and authorization** is provided by OAuth2 and OpenID Connect, which provides protection against unauthorized access.
- **Data encryption** at all levels (transport and storage).
- **Protection against DDoS attacks** with WAF and specialized protection services.

Disaster recovery

- **Data backup:** Regular backups of critical data and database.
- **Recovery plans:** Have clear plans in place to quickly restore system operations after disruptions, including the use of backup data centers and emergency sites.

Compromises:

- Balancing latency and consistency: With high speed requirements, it may be necessary to reduce the level of consistency.
- The cost of scaling: Increasing the number of services and databases can increase infrastructure and administration costs.