# System Design — Live Quiz Platform (50,000 Concurrent Players)

## 1. Core Components

- API Gateway / Load Balancer → routes player traffic to backend servers.
- WebSocket Servers → maintain real-time connections with players.
- Quiz Service → manages quiz creation, scheduling, and question delivery.
- Answer Service → receives answers from players and pushes them for processing.
- Leaderboard Service → keeps scores updated and provides real-time leaderboard.
- Database (SQL: MySQL/Postgres) → stores users, quizzes, questions, answers, scores.
- Cache (Redis) → fast in-memory storage for active sessions, quiz state, and leaderboard.
- Message Queue (Kafka/RabbitMQ) → handles high-volume answer events and distributes them.

## 2. Database Schema (Simplified)

- Users: id, name, email
- Quizzes: id, title, start_time, duration
- Questions: id, quiz_id, text, options_json, correct_option
- Enrollments: id, user_id, quiz_id, joined_at
- Answers: id, user_id, question_id, selected_option, timestamp
- Scores: user_id, quiz_id, total_score

## 3. Caching

- Redis stores: current quiz state, leaderboard, and player session data.
- Why: Redis provides sub-millisecond response times, necessary for real-time updates.

## 4. Message Queue

- Kafka / RabbitMQ collects player answer submissions.
- Decouples receiving answers from scoring.
- Scoring workers consume from queue, update Redis leaderboard, and persist results to DB.

## 5. Scaling Approach

- Horizontal scaling of stateless services behind load balancer.
- WebSocket sharding by quiz_id or player_id.
- Redis cluster for leaderboard performance.
- Database replicas for read-heavy operations.
- CDN for static quiz assets (images, etc.).

## 6. Simple Flow

- 1) Player connects to WebSocket server.
- 2) Quiz Service pushes question to players.
- 3) Player submits answer → Answer Service → Message Queue.
- 4) Scoring Worker consumes message → updates Redis → saves to DB.
- 5) Leaderboard Service reads Redis → WS pushes updates to players.

## 7. One-Line Summary

- Use WebSockets for real-time connections, Redis for fast leaderboard updates,
- a message queue for answer events, and SQL DB for persistence — scaled horizontally.