Interaktives Quiz-System mit Client/Server-Architektur

Ihr entwickelt ein netzwerkbasiertes Client/Server-Quizsystem, mit dem ein Ausbilder ein interaktives Live-Quiz starten kann. Teilnehmende (Auszubildende) verbinden sich über IP und Zugangscode mit dem Server(per Handy oder PC) und nehmen live am Quiz teil.

Technische Anforderungen

Verbindung & Kommunikation

- Aufbau einer **Client/Server-Verbindung über IP-Adresse**
- Zusätzlich wird ein **Zugangscode** erzeugt (z. B. 6-stellig), um sich zum aktuellen Quiz zu verbinden

Server (Quiz-Master-Seite)

- Benutzeroberfläche für den Ausbilder:
  - **Start eines neuen Quiz**
  - Auswahl eines **Themengebiets** (z. B. "Datenbanknormalisierung")
  - Optional: Möglichkeit, einzelne Fragen **vor dem Start zu deaktivieren**
  - Startet dann das Quiz
- Während einer aktiven Frage:
  - Anzeige: **Wie viele Teilnehmende** bereits geantwortet haben (aber **keine Ergebnisse**!)
- Nach Ablauf/Zeitlimit/„Weiter"-Klick:
  - Anzeige der **richtigen Antwort**
  - Möglichkeit, **zur nächsten Frage** zu springen

Client (Auszubildende)

- Plattformunabhängiger Zugriff (Handy und PC)
- Eingabe der **IP-Adresse + Zugangscode**
- Anzeige der aktuellen Frage mit **vier Antwortmöglichkeiten**
- **Einmalige Abgabe der Antwort** pro Frage
- Anzeige, dass Antwort gespeichert wurde (aber keine Auswertung!)

Modul	Beschreibung
Server-App	GUI/CLI zur Steuerung des Quiz, Verwaltung der Fragen, Anzeigen von Statistiken
Client-App	GUI oder mobile Ansicht (z. B. Web-Oberfläche oder Konsolenclient)
Datenbank	Speicherung von Quizfragen, nach **Kategorien sortiert**

Technologien / Sprachen (frei wählbar)

- Programmiersprache nach Wahl:
  - z. B. Python (z. B. Flask, socket), C# (.NET), Node.js (mit Prisma/Express)

- **Datenbank**: SQLite, PostgreSQL oder MongoDB
- **Optionale Frontend-Tools**: React, HTML/CSS/JS, mobile Frameworks (z. B. Ionic)

Projektziele

Am Ende soll ein funktionierender Prototyp bestehen mit:

- Server-App mit Benutzeroberfläche für das Quiz-Management
- Mindestens **10 Quizfragen in zwei Kategorien**
- Verbindung von **mindestens zwei Clients gleichzeitig**
- Saubere Darstellung der Antwortverteilung am Server
- Klarer Abschluss der einzelnen Fragen und Übergang zur nächsten

Hinweise für die Umsetzung

- Die Gruppe **kann** sich in **Frontend-/Backend-/Datenbankteams** aufteilen
- Optional: Nutzt Tools wie Postman(API), Docker oder Git für Organisation und Entwicklung
- Achtet auf saubere Code-Struktur und sinnvolle Kommunikation über Netzwerk

Mögliche Erweiterungen (optional, nicht verpflichtend)

- Timer-Funktion pro Frage
- (Sicherer-)Login für den Ausbilder
- Live-Statistiken während der Session
- Frageimport über CSV/JSON
- Deployment auf einem Raspberry Pi oder eurem (V-)Server

Abgabe / Präsentation

- Live-Demonstration des Systems
- Kurze Dokumentation (PDF oder Readme) mit Beschreibung des Aufbaus
- Quellcode (z. B. per GitHub oder ZIP)
- Vorstellung des Aufbaus in einer kurzen Präsentation (5–10 Minuten)

Hilfestellungen

- Nutze das Dokument ***Quiz-App-Demo.pdf***
- Es gibt ein Beispielprojekt unter:
  - https://github.com/ChristopherSchwerdt/FIAE/tree/main/Projekte/Quiz-App
- Nutze das Dokument ***Dokumentation Quiz-App,pdf***