Online Quiz App Users can Join Individual User Can Join with Collaborative Every Quiz has Cash rewards and entry fee Every Question have credit points In the Quiz time has call option, 50-50 removal option and Need Support but every option is payable

##### **You said:**

need a architech for the same

###### **ChatGPT said:**

For your **Online Quiz App**, you need an architecture that supports:

1. **User Management** (Individual & Collaborative play)
2. **Quiz & Question Management** (Entry fees, rewards, points system)
3. **Payment Integration** (For quiz entry & payable options)
4. **Real-time Communication** (For collaboration & call options)
5. **Scalability & Performance** (Handling multiple concurrent users)

### **Proposed Architecture**

#### **1. Frontend (React + TypeScript)**

* **User Interface**: React with Next.js for SSR & fast loading
* **State Management**: Redux or React Query
* **WebSockets**: For real-time updates in collaborative mode

#### **2. Backend (Node.js + Express/NestJS)**

* **Authentication**: JWT or OAuth
* **Quiz Engine**: Manages questions, user progress, and scoring
* **Payments**: Stripe/Razorpay integration for transactions
* **Real-time Features**: WebSockets (Socket.io) for call options and collaboration
* **AI/ML for Question Difficulty**: Adaptive quiz difficulty based on user performance

#### **3. Database**

* **Primary DB**: PostgreSQL (Relational, structured data like quizzes, users, payments)
* **NoSQL**: MongoDB/DynamoDB (For fast retrieval of quiz history & leaderboard)
* **Caching**: Redis (To store frequently accessed data)

#### **4. Infrastructure (Cloud-Based - AWS/GCP/Azure)**

* **EC2 / Lambda**: For scalable backend services
* **RDS / DynamoDB**: Managed database services
* **S3 / CloudFront**: For storing quiz images, avatars, etc.
* **WebSockets**: AWS AppSync / Firebase for real-time interactions
* **CI/CD**: GitHub Actions for automated deployments

#### **5. Security & Compliance**

* **DDoS Protection**: AWS Shield / Cloudflare
* **Data Encryption**: AES-256 for sensitive user data
* **Compliance**: GDPR, PCI-DSS for payment processing

Would you like me to create a **detailed architecture diagram** or a **database schema** for this?

##### **You said:**

database schema all track and delta with raltions

###### **ChatGPT said:**

Here’s a well-structured **database schema** for your **Online Quiz App**, considering:

* **Tracking all user actions** (entry, progress, payments, support options used, etc.)
* **Delta changes** (maintaining history/versioning where needed)
* **Relations & indexing** (optimizing query performance)

### **Entities & Relationships**

#### **1. Users Table**

Stores user information.

sql

CopyEdit

CREATE TABLE users (

user\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

name VARCHAR(100) NOT NULL,

email VARCHAR(255) UNIQUE NOT NULL,

password\_hash TEXT NOT NULL,

balance DECIMAL(10,2) DEFAULT 0.00,

created\_at TIMESTAMP DEFAULT NOW(),

updated\_at TIMESTAMP DEFAULT NOW()

);

#### **2. Quizzes Table**

Stores quiz details.

sql

CopyEdit

CREATE TABLE quizzes (

quiz\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

title VARCHAR(255) NOT NULL,

entry\_fee DECIMAL(10,2) NOT NULL,

cash\_reward DECIMAL(10,2) NOT NULL,

created\_at TIMESTAMP DEFAULT NOW()

);

#### **3. Questions Table**

Stores quiz questions.

sql

CopyEdit

CREATE TABLE questions (

question\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

quiz\_id UUID REFERENCES quizzes(quiz\_id) ON DELETE CASCADE,

question\_text TEXT NOT NULL,

correct\_option VARCHAR(255) NOT NULL,

points INT NOT NULL,

created\_at TIMESTAMP DEFAULT NOW()

);

#### **4. Options Table**

Stores multiple-choice options for each question.

sql

CopyEdit

CREATE TABLE options (

option\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

question\_id UUID REFERENCES questions(question\_id) ON DELETE CASCADE,

option\_text VARCHAR(255) NOT NULL,

is\_correct BOOLEAN DEFAULT FALSE

);

#### **5. User Quiz Participation Table**

Tracks which users join which quizzes.

sql

CopyEdit

CREATE TABLE user\_quiz\_participation (

participation\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

user\_id UUID REFERENCES users(user\_id) ON DELETE CASCADE,

quiz\_id UUID REFERENCES quizzes(quiz\_id) ON DELETE CASCADE,

score INT DEFAULT 0,

amount\_paid DECIMAL(10,2) NOT NULL,

joined\_at TIMESTAMP DEFAULT NOW()

);

#### **6. User Answers Table**

Tracks which answer a user selects for a question.

sql

CopyEdit

CREATE TABLE user\_answers (

answer\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

participation\_id UUID REFERENCES user\_quiz\_participation(participation\_id) ON DELETE CASCADE,

question\_id UUID REFERENCES questions(question\_id) ON DELETE CASCADE,

selected\_option\_id UUID REFERENCES options(option\_id) ON DELETE CASCADE,

is\_correct BOOLEAN,

points\_awarded INT DEFAULT 0,

answered\_at TIMESTAMP DEFAULT NOW()

);

#### **7. Payment Transactions Table**

Stores all financial transactions (entry fees, support options, cash rewards).

sql

CopyEdit

CREATE TABLE transactions (

transaction\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

user\_id UUID REFERENCES users(user\_id) ON DELETE CASCADE,

amount DECIMAL(10,2) NOT NULL,

transaction\_type VARCHAR(50) CHECK (transaction\_type IN ('entry\_fee', 'reward', 'support\_option')),

status VARCHAR(20) CHECK (status IN ('success', 'failed', 'pending')),

created\_at TIMESTAMP DEFAULT NOW()

);

#### **8. Support Options Used Table**

Tracks the usage of payable support options (Call, 50-50, Need Support).

sql

CopyEdit

CREATE TABLE support\_options\_used (

usage\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

participation\_id UUID REFERENCES user\_quiz\_participation(participation\_id) ON DELETE CASCADE,

question\_id UUID REFERENCES questions(question\_id) ON DELETE CASCADE,

option\_used VARCHAR(50) CHECK (option\_used IN ('call', '50-50', 'need\_support')),

amount\_paid DECIMAL(10,2) NOT NULL,

used\_at TIMESTAMP DEFAULT NOW()

);

#### **9. Leaderboard Table**

Maintains rankings based on scores.

sql

CopyEdit

CREATE TABLE leaderboard (

leaderboard\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

quiz\_id UUID REFERENCES quizzes(quiz\_id) ON DELETE CASCADE,

user\_id UUID REFERENCES users(user\_id) ON DELETE CASCADE,

total\_score INT NOT NULL,

rank INT NOT NULL,

updated\_at TIMESTAMP DEFAULT NOW()

);

#### **10. Audit Logs Table**

Tracks changes in key tables for historical tracking.

sql

CopyEdit

CREATE TABLE audit\_logs (

log\_id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

table\_name VARCHAR(100) NOT NULL,

record\_id UUID NOT NULL,

action VARCHAR(50) CHECK (action IN ('INSERT', 'UPDATE', 'DELETE')),

previous\_data JSONB,

new\_data JSONB,

changed\_at TIMESTAMP DEFAULT NOW()

);

### **Indexes & Optimizations**

**Indexes for fast retrieval** sql  
CopyEdit  
CREATE INDEX idx\_user\_email ON users(email);

CREATE INDEX idx\_quiz\_title ON quizzes(title);

CREATE INDEX idx\_participation\_user ON user\_quiz\_participation(user\_id);

CREATE INDEX idx\_transaction\_user ON transactions(user\_id);

**Partitioning for Audit Logs** (Monthly partitioning for performance)  
  
 sql  
CopyEdit  
CREATE TABLE audit\_logs\_2025\_03 PARTITION OF audit\_logs

FOR VALUES FROM ('2025-03-01') TO ('2025-04-01');

### **Tracking & Delta Changes**

1. **Audit Logs** capture inserts, updates, and deletes.

**Versioning for Questions** (optional)  
 sql  
CopyEdit  
ALTER TABLE questions ADD COLUMN version INT DEFAULT 1;

### **Next Steps**

* Would you like an **ER Diagram**?
* Need API endpoints for integration?
* Want me to include **Redis caching** for leaderboard or active quizzes?

**Online Quiz App - Full Architecture & Database Schema**

## **1. System Architecture**

### **Frontend (Client-Side)**

* **Web App:** React.js / Next.js
* **Mobile App:** React Native / Flutter
* **Communication:** WebSockets & REST API

### **Backend (Server-Side)**

* **Framework:** Node.js with Express / FastAPI (Python)
* **Real-Time Processing:** WebSockets (Socket.io) / AWS AppSync (GraphQL)
* **Authentication:** JWT-based authentication

### **Database (Storage & Caching)**

* **Primary DB:** PostgreSQL / MySQL (Structured Data) OR MongoDB / DynamoDB (Flexible Schema)
* **Cache:** Redis for leaderboards, active users, and sessions
* **Event Streaming:** Kafka / AWS Kinesis for live updates

### **Infrastructure & Deployment**

* **Auto-Scaling:** AWS Lambda (Serverless) OR AWS EC2 with Kubernetes
* **Content Delivery:** AWS CloudFront / CDN for optimized performance
* **Load Balancing:** AWS ALB / Nginx

## **2. Database Schema Design (SQL)**

### **1. Users Table**

****CREATE TABLE users (

user\_id UUID PRIMARY KEY,

name VARCHAR(255) NOT NULL,

email VARCHAR(255) UNIQUE NOT NULL,

phone VARCHAR(20) UNIQUE,

wallet\_balance DECIMAL(10,2) DEFAULT 0.00,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

### **2. Quiz Table**

****CREATE TABLE quizzes (

quiz\_id UUID PRIMARY KEY,

title VARCHAR(255) NOT NULL,

entry\_fee DECIMAL(10,2) NOT NULL,

reward\_pool DECIMAL(10,2) NOT NULL,

max\_participants INT NOT NULL,

start\_time TIMESTAMP NOT NULL,

end\_time TIMESTAMP NOT NULL,

status ENUM('upcoming', 'ongoing', 'completed') NOT NULL

);

### **3. Questions Table**

****CREATE TABLE questions (

question\_id UUID PRIMARY KEY,

quiz\_id UUID REFERENCES quizzes(quiz\_id) ON DELETE CASCADE,

question\_text TEXT NOT NULL,

option\_a TEXT NOT NULL,

option\_b TEXT NOT NULL,

option\_c TEXT NOT NULL,

option\_d TEXT NOT NULL,

correct\_answer CHAR(1) NOT NULL,

points INT NOT NULL

);

### **4. Participants Table**

****CREATE TABLE participants (

participant\_id UUID PRIMARY KEY,

user\_id UUID REFERENCES users(user\_id) ON DELETE CASCADE,

quiz\_id UUID REFERENCES quizzes(quiz\_id) ON DELETE CASCADE,

score INT DEFAULT 0,

rank INT,

status ENUM('active', 'disqualified', 'won') NOT NULL

);

### **5. Transactions Table**

****CREATE TABLE transactions (

transaction\_id UUID PRIMARY KEY,

user\_id UUID REFERENCES users(user\_id) ON DELETE CASCADE,

quiz\_id UUID REFERENCES quizzes(quiz\_id) ON DELETE CASCADE,

amount DECIMAL(10,2) NOT NULL,

transaction\_type ENUM('entry\_fee', 'reward') NOT NULL,

timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

### **6. Live Quiz Tracking (Leaderboards & Status)**

****CREATE TABLE quiz\_tracking (

tracking\_id UUID PRIMARY KEY,

quiz\_id UUID REFERENCES quizzes(quiz\_id) ON DELETE CASCADE,

current\_question UUID REFERENCES questions(question\_id),

active\_users JSONB,

leaderboard JSONB

);

### **7. Fraud Monitoring & AI-based Cheating Detection**

****CREATE TABLE fraud\_logs (

log\_id UUID PRIMARY KEY,

user\_id UUID REFERENCES users(user\_id) ON DELETE CASCADE,

quiz\_id UUII want SQL DB schema for that sameD REFERENCES quizzes(quiz\_id) ON DELETE CASCADE,

suspicious\_activity BOOLEAN NOT NULL,

reason TEXT,

timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);



## **3. Key Features & Business Logic**

✅ **Large Pool Participation:** Auto-scalable backend to handle 10,000+ players per quiz.  
✅ **Real-time Quiz Engine:** WebSockets for live question sync & leaderboard updates.  
✅ **Payment Handling:** Integrated wallet system for transactions.  
✅ **Automated Rewards Distribution:** Score-based rankings decide rewards.  
✅ **Fraud Prevention:** AI-based cheating detection for fair gameplay.

This architecture and SQL database schema ensure scalability, security, and real-time engagement for a seamless online quiz experience. Let me know if you need further refinements! 🚀