

NEET CHAMP (Quiz for NEET online Prototype)

Quizizz learning app - Model

1. Front-End (User Interface)

- **Technology Stack:** React.js, Next.js, or Angular for web, Flutter or React Native for mobile apps.
- **UI/UX:** Engaging and gamified design with animations, leaderboards, and avatars.
- **User Roles:**
 - **Students:** Take quizzes, track progress, earn points.
 - **Teachers:** Create, assign, and analyze quizzes.
 - **Admins:** Manage content, users, and settings.
- **Features:**
 - Live quiz mode & Homework mode.
 - Question timers and power-ups.
 - Theme customization.
 - Reports & Analytics Dashboard.

2. Back-End (Server-Side)

- **Technology Stack:** Node.js with Express.js (or Django for Python-based backend).
- **APIs:** RESTful or GraphQL APIs to handle quiz data, authentication, leaderboards, etc.
- **Microservices:** Separate services for quiz management, scoring, and reporting.
- **Authentication & Authorization:**
 - OAuth, Firebase Auth, JWT tokens for login/signup.
 - Google/Microsoft account integration for school logins.
- **WebSockets:** Used for real-time interactions like live quizzes.

3. Database (Storage System)

- **Primary Database:** MongoDB (NoSQL) or PostgreSQL (SQL) for quiz, user, and session storage.
- **Real-time Database:** Firebase Firestore for syncing quiz sessions.
- **Caching:** Redis for storing live quiz scores and session data.

4. Quiz & Content Management System (CMS)

- **Question Bank:** Allows teachers to create, store, and share quizzes.
- **AI-powered Question Generation:** NLP models (like GPT) to generate questions automatically.
- **Multimedia Support:** Images, videos, and audio in questions.
- **Gamification Features:** Power-ups, XP points, streaks.

5. AI & Machine Learning (Personalization & Analytics)

- **Recommendation System:** Suggests quizzes based on previous performance.

- **Adaptive Learning:** AI adjusts difficulty levels based on student progress.
- **Performance Analytics:** Tracks student engagement, weak areas, and strengths.
- **Chatbot Support:** AI-powered chatbot to assist students and teachers.

6. Deployment & Hosting

- **Cloud Services:** AWS (S3 for storage, Lambda for serverless functions, EC2 for hosting).
- **CDN (Content Delivery Network):** Cloudflare or AWS CloudFront for faster quiz access.
- **CI/CD (Continuous Integration & Deployment):** GitHub Actions, Jenkins for auto-deployment.

7. Security & Compliance

- **Data Encryption:** AES-256 encryption for user data.
- **Privacy Laws Compliance:** COPPA, GDPR, FERPA compliance for student data.
- **DDoS Protection:** Cloudflare for security against cyber threats.

✂ API Structure for Quizizz Model

A Quizizz-like system involves several components:

1. **User Management** (Authentication, Profiles)
2. **Quiz Creation & Management**
3. **Question Bank & Categories**
4. **Live Quiz & Multiplayer Mode**
5. **Leaderboards & Analytics**
6. **Results & Reports**
7. **Payments & Subscription (optional)**

✂ API Endpoints:

1 Authentication & User Management

- **POST /api/auth/register** → User Signup
- **POST /api/auth/login** → Login with email/password
- **POST /api/auth/logout** → Logout
- **GET /api/user/profile** → Get user details
- **PUT /api/user/update** → Update profile

2 Quiz Creation & Management

- **POST /api/quiz/create** → Create a new quiz
- **GET /api/quiz/{quizId}** → Fetch quiz details
- **PUT /api/quiz/{quizId}/update** → Update quiz info
- **DELETE /api/quiz/{quizId}/delete** → Delete quiz

3 Question Bank & Categories

- **POST /api/question/add** → Add a new question
- **GET /api/question/{questionId}** → Fetch question details
- **PUT /api/question/{questionId}/update** → Update question
- **DELETE /api/question/{questionId}/delete** → Delete question
- **GET /api/categories** → Get all quiz categories

4 Live Quiz & Multiplayer Mode

- **POST /api/quiz/{quizId}/start** → Start a quiz session
- **POST /api/quiz/{quizId}/join** → Join a live quiz
- **POST /api/quiz/{quizId}/answer** → Submit an answer
- **GET /api/quiz/{quizId}/leaderboard** → View current rankings

5 Leaderboards & Analytics

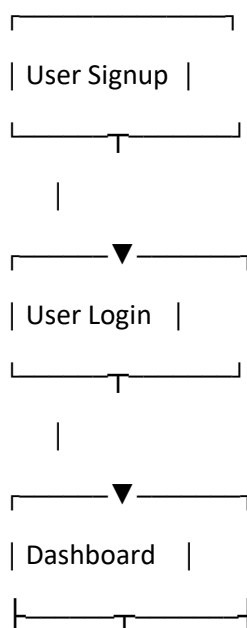
- **GET /api/leaderboard/global** → Get global rankings
- **GET /api/leaderboard/{quizId}** → Get leaderboard for a specific quiz
- **GET /api/user/stats** → Fetch user performance stats

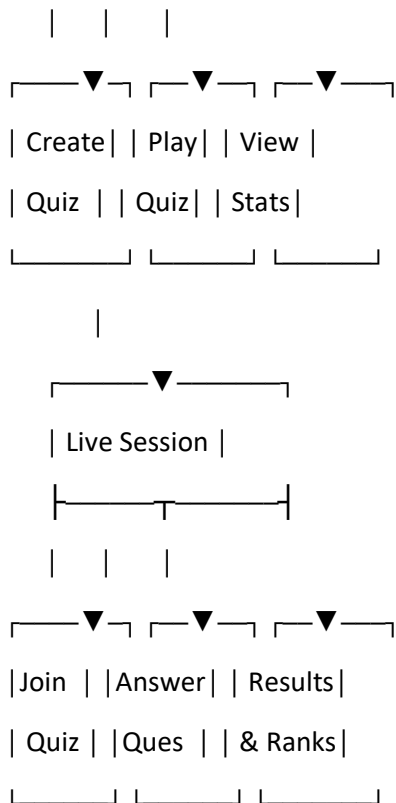
6 Results & Reports

- **GET /api/quiz/{quizId}/results** → Fetch quiz results
- **GET /api/user/results** → Fetch user's past quiz results
- **POST /api/report/generate** → Generate a detailed performance report

7 Payments & Subscription (if needed)

- **POST /api/subscribe** → Subscribe to premium
- **GET /api/subscription/status** → Check subscription status





A detailed database schema for **Quizizz** (or a similar quiz-based platform) would include various tables to handle users, quizzes, questions, responses, leaderboards, and more. Below is a detailed breakdown:

1. Users Table (`users`)

Stores user information, including teachers, students, and admins.

Column Name	Data Type	Constraints	Description
<code>user_id</code>	UUID / INT	PRIMARY KEY, AUTO_INCREMENT	Unique ID for each user
<code>name</code>	VARCHAR(255)	NOT NULL	Full name of the user
<code>email</code>	VARCHAR(255)	UNIQUE, NOT NULL	User's email
<code>password_hash</code>	VARCHAR(255)	NOT NULL	Hashed password
<code>role</code>	ENUM('student', 'teacher', 'admin')	NOT NULL	Role of the user

Column Name	Data Type	Constraints	Description
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Account creation date
updated_at	TIMESTAMP	ON UPDATE CURRENT_TIMESTAMP	Last updated timestamp

2. Quizzes Table (`quizzes`)

Stores quizzes created by teachers.

Column Name	Data Type	Constraints	Description
quiz_id	UUID / INT	PRIMARY KEY, AUTO_INCREMENT	Unique quiz ID
title	VARCHAR(255) NOT NULL		Quiz title
description	TEXT	NULL	Quiz description
created_by	UUID / INT	FOREIGN KEY (<code>users.user_id</code>)	Creator of the quiz
is_public	BOOLEAN	DEFAULT TRUE	Whether quiz is public or private
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Creation date
updated_at	TIMESTAMP	ON UPDATE CURRENT_TIMESTAMP	Last update timestamp

3. Questions Table (`questions`)

Stores questions related to quizzes.

Column Name	Data Type	Constraints	Description
question_id	UUID / INT	PRIMARY KEY, AUTO_INCREMENT	Unique question ID
quiz_id	UUID / INT	FOREIGN KEY (<code>quizzes.quiz_id</code>)	Associated quiz ID
question_text	TEXT	NOT NULL	Question content
question_type	ENUM('MCQ', 'True/False', 'Fill in the Blank', 'Short Answer')	NOT NULL	Type of question

Column Name	Data Type	Constraints	Description
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Creation timestamp

4. Options Table (`options`)

Stores answer choices for multiple-choice questions.

Column Name	Data Type	Constraints	Description
option_id	UUID / INT	PRIMARY KEY, AUTO_INCREMENT	Unique option ID
question_id	UUID / INT	FOREIGN KEY (questions.question_id)	Associated question ID
option_text	TEXT	NOT NULL	Answer option text
is_correct	BOOLEAN	DEFAULT FALSE	Marks correct option

5. Responses Table (`responses`)

Stores student responses.

Column Name	Data Type	Constraints	Description
response_id	UUID / INT	PRIMARY KEY, AUTO_INCREMENT	Unique response ID
user_id	UUID / INT	FOREIGN KEY (users.user_id)	Student who answered
question_id	UUID / INT	FOREIGN KEY (questions.question_id)	Question being answered
option_id	UUID / INT	FOREIGN KEY (options.option_id)	Selected option ID
is_correct	BOOLEAN	DEFAULT FALSE	Whether answer was correct
submitted_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Submission time

6. Leaderboard Table (`leaderboard`)

Stores leaderboard rankings based on quiz performance.

Column Name	Data Type	Constraints	Description
leaderboard_id	UUID / INT	PRIMARY KEY, AUTO_INCREMENT	Unique ID
quiz_id	UUID / INT	FOREIGN KEY (quizzes.quiz_id)	Associated quiz ID
user_id	UUID / INT	FOREIGN KEY (users.user_id)	Student's ID
score	INT	NOT NULL	Score achieved
rank	INT	NOT NULL	Ranking position

7. Quiz Sessions Table (`quiz_sessions`)

Stores live quiz sessions.

Column Name	Data Type	Constraints	Description
session_id	UUID / INT	PRIMARY KEY, AUTO_INCREMENT	Unique session ID
quiz_id	UUID / INT	FOREIGN KEY (quizzes.quiz_id)	Associated quiz ID
hosted_by	UUID / INT	FOREIGN KEY (users.user_id)	Teacher hosting session
start_time	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Session start time
end_time	TIMESTAMP	NULL	Session end time

8. Reports Table (`reports`)

Stores reports of quiz performance.

Column Name	Data Type	Constraints	Description
report_id	UUID / INT	PRIMARY KEY, AUTO_INCREMENT	Unique report ID
quiz_id	UUID / INT	FOREIGN KEY (quizzes.quiz_id)	Quiz associated with report
user_id	UUID / INT	FOREIGN KEY (users.user_id)	Student's ID
total_questions	INT	NOT NULL	Number of questions attempted
correct_answers	INT	NOT NULL	Correctly answered questions

Column Name	Data Type	Constraints	Description
score	INT	NOT NULL	Final score
generated_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Report generated timestamp

9. Tags Table (`tags`)

Stores tags for quizzes (e.g., "Math", "Science", "History").

Column Name	Data Type	Constraints	Description
tag_id	UUID / INT	PRIMARY KEY, AUTO_INCREMENT	Unique tag ID
tag_name	VARCHAR(255)	UNIQUE, NOT NULL	Name of the tag

10. Quiz Tags Table (`quiz_tags`)

Associates quizzes with tags.

Column Name	Data Type	Constraints	Description
quiz_id	UUID / INT	FOREIGN KEY (<code>quizzes.quiz_id</code>)	Quiz ID
tag_id	UUID / INT	FOREIGN KEY (<code>tags.tag_id</code>)	Tag ID
PRIMARY KEY (<code>quiz_id</code> , <code>tag_id</code>)			

Relationships Between Tables

- `users` → `quizzes` (One-to-Many) → One teacher can create multiple quizzes.
- `quizzes` → `questions` (One-to-Many) → A quiz can have multiple questions.
- `questions` → `options` (One-to-Many) → A question can have multiple answer choices.
- `questions` → `responses` (One-to-Many) → One question can have multiple responses.
- `users` → `responses` (One-to-Many) → One user can submit multiple responses.
- `quizzes` → `quiz_sessions` (One-to-Many) → A quiz can have multiple sessions.
- `users` → `leaderboard` (One-to-Many) → One user can appear on multiple leaderboards.
- `quizzes` → `quiz_tags` (Many-to-Many) → A quiz can have multiple tags.

This schema allows **efficient data retrieval** for:

- Tracking **user progress**

- Generating **leaderboards & reports**
- Managing **live quiz sessions**
- Storing and analyzing **quiz performance**

Would you like to add any additional features like **AI-based difficulty scaling, time tracking**, or **question banks**? 🚀

4o

You said:

AI-based difficulty scaling, time tracking, or question banks? 🚀

ChatGPT said:

Great choices! Let's expand the schema to include **AI-based difficulty scaling, time tracking, and question banks**. 🚀

11. Question Bank Table (`question_bank`)

Stores a pool of categorized questions that can be reused in different quizzes.

Column Name	Data Type	Constraints	Description
<code>bank_id</code>	UUID / INT	PRIMARY KEY, AUTO_INCREMENT	Unique question bank ID
<code>title</code>	VARCHAR(255)	NOT NULL	Name of the question bank
<code>created_by</code>	UUID / INT	FOREIGN KEY (<code>users.user_id</code>)	Creator of the question bank
<code>is_public</code>	BOOLEAN	DEFAULT TRUE	Visibility of the bank
<code>created_at</code>	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Creation date

12. Bank Questions Table (`bank_questions`)

Stores questions associated with a question bank.

Column Name	Data Type	Constraints	Description
<code>bank_question_id</code>	UUID / INT	PRIMARY KEY, AUTO_INCREMENT	Unique question ID
<code>bank_id</code>	UUID / INT	FOREIGN KEY (<code>question_bank.bank_id</code>)	Associated question bank

Column Name	Data Type	Constraints	Description
question_text	TEXT	NOT NULL	Question content
difficulty_level	ENUM('easy', 'medium', 'hard')	NOT NULL	Predefined difficulty level
question_type	ENUM('MCQ', 'True/False', 'Fill in the Blank', 'Short Answer')	NOT NULL	Type of question
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Creation timestamp

13. AI Difficulty Scaling Table (`ai_difficulty`)

Stores historical response data to dynamically adjust question difficulty.

Column Name	Data Type	Constraints	Description
question_id	UUID / INT	FOREIGN KEY (questions.question_id)	Question ID
user_id	UUID / INT	FOREIGN KEY (users.user_id)	User who answered
quiz_id	UUID / INT	FOREIGN KEY (quizzes.quiz_id)	Associated quiz
difficulty_level	ENUM('easy', 'medium', 'hard')	NOT NULL	Difficulty level at the time of answering
is_correct	BOOLEAN	NOT NULL	Whether the answer was correct
time_taken	INT	NOT NULL (milliseconds)	Time taken to answer (in ms)
timestamp	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Submission timestamp

How AI Scaling Works?

- **Increase difficulty:** If many users answer a question correctly and quickly, its difficulty level increases.
- **Decrease difficulty:** If most users get it wrong or take too long, the difficulty level drops.
- **Adaptive learning:** AI selects future questions based on previous performance.

14. Time Tracking Table (time_tracking)

Logs the time taken by users for each question and quiz.

Column Name	Data Type	Constraints	Description
time_id	UUID / INT	PRIMARY KEY, AUTO_INCREMENT	Unique time tracking ID
user_id	UUID / INT	FOREIGN KEY (users.user_id)	User attempting the quiz
quiz_id	UUID / INT	FOREIGN KEY (quizzes.quiz_id)	Associated quiz
question_id	UUID / INT	FOREIGN KEY (questions.question_id)	Question ID
time_taken	INT	NOT NULL (milliseconds)	Time taken to answer
submitted_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Submission time

Use Cases

- Helps analyze which questions take the longest.
- Allows **AI to adjust difficulty** based on response time.
- Provides insights into **user behavior and engagement**.

New Relationships Between Tables

- **question_bank** → **bank_questions** (*One-to-Many*)
- **bank_questions** → **questions** (*One-to-Many*) (A quiz question may originate from a bank)
- **ai_difficulty** → **questions** (*One-to-One*) (Each question's difficulty can be analyzed)
- **time_tracking** → **responses** (*One-to-One*) (Each answer has a recorded time)

Final Benefits of These Additions

- ✓ **Question Banks** → Reuse quality questions & ensure consistency.
- ✓ **AI Scaling** → Adapts quizzes to match students' skill levels.
- ✓ **Time Tracking** → Helps optimize quizzes for better engagement.

Would you like to integrate **gamification (badges, streaks, etc.)** next? 🏆🔧

4o

You said:

gamification (badges, streaks, etc.)

ChatGPT said:

Gamification System for Quizizz 🎮🚀

Adding **gamification** can make quizzes **engaging & motivating** for users! Here's how we can implement **badges, streaks, XP, levels, and achievements**.

15. Experience & Levels Table (`experience`)

Tracks user experience points (XP) and levels.

Column Name	Data Type	Constraints	Description
<code>xp_id</code>	UUID / INT	PRIMARY KEY, AUTO_INCREMENT	Unique XP ID
<code>user_id</code>	UUID / INT	FOREIGN KEY (<code>users.user_id</code>)	Associated user
<code>total_xp</code>	INT	DEFAULT 0	Total XP collected
<code>level</code>	INT	DEFAULT 1	Current user level
<code>last_updated</code>	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Last XP update

XP Calculation Rules

- **+10 XP** for every correct answer.
 - **+5 XP** for attempting a question.
 - **+50 XP** for winning a quiz session.
 - **+20 XP** for maintaining a daily streak.
-

16. Streaks Table (`streaks`)

Tracks consecutive days a user participates in quizzes.

Column Name	Data Type	Constraints	Description
<code>streak_id</code>	UUID / INT	PRIMARY KEY, AUTO_INCREMENT	Unique streak ID
<code>user_id</code>	UUID / INT	FOREIGN KEY (<code>users.user_id</code>)	Associated user
<code>current_streak</code>	INT	DEFAULT 0	Active streak count (days)
<code>longest_streak</code>	INT	DEFAULT 0	User's highest streak
<code>last_activity</code>	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Last time user played

Streak System

- **Daily Quiz Completion = +1 Streak Day**
- **Skipping a Day = Reset to 0**
- **Bonus XP for Streaks** (e.g., 50 XP for a 7-day streak)

17. Badges Table (badges)

Stores all possible badges.

Column Name	Data Type	Constraints	Description
badge_id	UUID / INT	PRIMARY KEY, AUTO_INCREMENT	Unique badge ID
badge_name	VARCHAR(255)	UNIQUE, NOT NULL	Name of the badge
badge_desc	TEXT	NULL	Description of badge criteria
badge_icon	VARCHAR(255)	NULL	URL or path for badge image

Examples of Badges 🏆

Badge Name	Criteria
Quiz Master 🏆	Win 10 quizzes
Streak Keeper 🔥	Maintain a 7-day streak
Fast Thinker ⚡	Answer 5 questions in <5 sec each
Top Scorer 🏅	Score 90%+ in 3 quizzes
Comeback King 🏰	Answer last 3 questions correctly after failing before

18. User Badges Table (user_badges)

Links badges to users.

Column Name	Data Type	Constraints	Description
user_id	UUID / INT	FOREIGN KEY (users.user_id)	User earning badge
badge_id	UUID / INT	FOREIGN KEY (badges.badge_id)	Earned badge

Column Name	Data Type	Constraints	Description
earned_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Date badge was awarded
PRIMARY KEY (user_id, badge_id)			

19. Leaderboard Enhancements (`leaderboard`)

Already defined, but now we add **XP-based ranking!**

Column Name	Data Type	Constraints	Description
rank	INT	NOT NULL	Position on leaderboard
score	INT	NOT NULL	Quiz performance score
xp_points	INT	NOT NULL	User XP (determines tie-breaker)

🏆 Now users are ranked by score & XP!

20. Achievements Table (`achievements`)

Stores milestone achievements (e.g., number of quizzes played).

Column Name	Data Type	Constraints	Description
achievement_id	UUID / INT	PRIMARY KEY, AUTO_INCREMENT	Unique achievement ID
user_id	UUID / INT	FOREIGN KEY (users.user_id)	Associated user
achievement_name	VARCHAR(255)	NOT NULL	Name of achievement
achievement_value	INT	NOT NULL	Number of times achieved
last_updated	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Last achievement update

Examples of Achievements 🏆

Achievement Name	Criteria
Quiz Enthusiast 🏆	Play 50 quizzes

Achievement Name	Criteria
Perfectionist ★	Score 100% in 5 quizzes
Speed Demon □	Answer 10 questions in <3 sec each
XP Collector 🏆	Earn 5000 XP

Final Benefits of Gamification 🎮🔥

- ✓ **Badges & Achievements** → Makes learning rewarding.
- ✓ **XP & Levels** → Encourages progress & competition.
- ✓ **Daily Streaks** → Boosts retention & engagement.
- ✓ **Enhanced Leaderboards** → Adds motivation & challenge.