Example prompt

ZapQuiz: Interactive Quiz Application

Part 1

Project Overview

ZapQuiz is a Kahoot-like interactive quiz application that allows teachers to create quizzes and participants to join via QR codes, answering questions in real-time with a points-based scoring system.

Technology Stack

- **Backend:** PHP
- **Frontend:** HTML, CSS, JavaScript (responsive design)
- **Database:** MySQL/MariaDB
- **CSS Framework:** Tailwind CSS
- **Real-time Updates:** SSE (Server-Snet events)
- **Installation:** Must support subdirectory installation (e.g., https://www.server.com/zapquiz)

Development Phases

The project will be developed in three distinct phases:

1. **Teacher Authentication & Quiz Management** (current focus)
2. Quiz Session Initialization & Participant Onboarding
3. Quiz Execution, Scoring & Results Display

Phase 1: Teacher Authentication & Quiz Management System

Requirements

User Authentication

1. Create a secure login system for teachers with:
   - Username/email and password authentication
   - Password encryption using bcrypt
   - Session management
   - Basic "remember me" functionality
   - Password reset capability (optional)

Database Schema (Initial)

1. **Teachers Table:**
   - id (PRIMARY KEY)
   - username
   - email
   - password (bcrypt hashed)
   - created_at
   - updated_at
2. **Quizzes Table:**
   - id (PRIMARY KEY)
   - teacher_id (FOREIGN KEY)
   - title
   - description
   - unique_identifier (for QR code generation)
   - status (draft, active, completed)

- created_at
- updated_at

3. **Questions Table:**
   - id (PRIMARY KEY)
   - quiz_id (FOREIGN KEY)
   - question_text (HTML-formatted)
   - image_path (optional)
   - display_order
   - time_limit (default: 30 seconds)
   - created_at
   - updated_at
4. **Answers Table:**
   - id (PRIMARY KEY)
   - question_id (FOREIGN KEY)
   - answer_text
   - is_correct (boolean)
   - answer_option (A, B, C, or D)
   - created_at
   - updated_at

Teacher Dashboard Functionality

1. **Authentication Pages:**
   - Login page
   - For now we don't need an registration page yet.
     Create one login with user *teacher*, password *teacher123*
   - No password recovery yet.
2. **Quiz Management:**
   - List all quizzes with search/filter options
   - Create new quiz (with auto-generated unique identifier)

- Edit existing quiz details
- Delete quiz (with confirmation)
- Duplicate quiz with the option to assign it to a different teacher.

3. **Question Management:**
   - List all questions for a quiz
   - Add new questions with:
     - Question text (HTML editor)
     - Image upload capability
     - Four answer options (A, B, C, D)
     - Correct answer designation
     - Question reordering capability
   - Edit existing questions
   - Delete questions (with confirmation)

Technical Implementation Details

1. **Directory Structure:**

```
zapquiz/
├── assets/
│   ├── css/
│   ├── js/
│   ├── images/
│   └── uploads/
├── config/
│   └── database.php
├── includes/
│   ├── auth.php
│   ├── functions.php
│   └── header.php
├── models/
│   ├── Teacher.php
│   ├── Quiz.php
│   └── Question.php
├── views/
```

```
│   ├── auth/
│   ├── dashboard/
│   └── quiz/
├── index.php
└── .htaccess (for subdirectory configuration)
```

2. **URL Structure:**
   - `/zapquiz/` - Landing page
   - `/zapquiz/login` - Teacher login
   - `/zapquiz/dashboard` - Teacher dashboard
   - `/zapquiz/quiz/create` - Create new quiz
   - `/zapquiz/quiz/{id}/edit` - Edit quiz
   - `/zapquiz/quiz/{id}/questions` - Manage questions
   - `/zapquiz/quiz/{id}/start` - Initialize quiz (Phase 2)
3. **Security Considerations:**
   - Input validation and sanitization
   - CSRF protection for forms
   - Session security
   - Proper handling of file uploads
   - Database query parameterization

UI/UX Requirements

1. **Teacher Dashboard:**
   - Clean, professional interface using Tailwind CSS
   - Responsive design for both desktop and tablet use
   - Intuitive navigation between quizzes and questions
   - Visual indication of quiz status (draft, active, completed)
2. **Quiz & Question Editor:**
   - User-friendly HTML editor for question text
   - Simple image upload with preview
   - Clear visual distinction between correct and incorrect answers

- Drag-and-drop functionality for question reordering (optional)

Deliverables for Phase 1

1. Functional teacher authentication system
2. Complete quiz CRUD operations
3. Question and answer management system
4. Responsive UI built with Tailwind CSS
5. Database with appropriate schema and relationships
6. Installation instructions for subdirectory setup

Next Steps After Phase 1

Upon completion of Phase 1, development will proceed to Phase 2, focusing on quiz initialization, QR code generation, and participant onboarding.

--

ZapQuiz - Quiz Initialization & Participant Onboarding Prompt

Flow Overview

For the second phase of the ZapQuiz application, we need to implement the quiz initialization flow and participant onboarding process. This system bridges the teacher dashboard to the actual quiz experience.

Requirements

Teacher Dashboard to Quiz Launch

1. Implement a "Start Quiz" button on the teacher dashboard for each available quiz

2. When clicked, this should navigate to a splash page designed for classroom display

Quiz Splash Page (Teacher View)

Design a splash screen that includes:

1. A prominent, large QR code that students can scan with their phones
   - The QR code should contain a unique URL for joining the specific quiz session
   - The URL should be in the format: `https://[domain]/zapquiz/join/[unique-code]`
2. A text version of the join link displayed beneath the QR code
   - Make this text clearly visible for situations where QR scanning isn't possible
   - Consider making this text easily selectable for sharing via other means
3. A real-time participant list
   - As students join, their names should appear in this list
   - Use Server-Sent Events or WebSockets to update the list without page refresh
   - Include a counter showing the total number of participants
4. A prominent "Begin Quiz" button
   - This should be enabled only when at least one participant has joined
   - Include a confirmation dialog to prevent accidental starts

Participant Join Flow (Student View)

When a student scans the QR code or enters the join link:

1. Create a mobile-friendly landing page with:
   - Quiz title
   - Teacher's name
   - A form to enter the student's name
   - Clear instructions
2. After name submission:
   - Validate the name (no empty submissions, reasonable length)
   - Store the participant in the database with a unique session ID

- Redirect to a waiting screen

3. Design a waiting screen that:
   - Confirms successful registration
   - Displays a friendly "Waiting for quiz to begin..." message
   - Includes a visual indicator that the connection is active
   - Uses Server-Sent Events or WebSockets to listen for quiz start signal

Database Requirements

1. Add a `sessions` table to track active quiz sessions:
   - session_id (PRIMARY KEY)
   - quiz_id (FOREIGN KEY)
   - status (waiting, active, completed)
   - started_at
   - ended_at
2. Add a `participants` table:
   - id (PRIMARY KEY)
   - session_id (FOREIGN KEY)
   - name
   - joined_at
   - current_score (default 0)
   - last_active

Real-Time Communication Requirements

1. Implement Server-Sent Events or WebSockets to:
   - Update the teacher's view with participant names as they join
   - Notify participants when the quiz begins
   - Maintain connection status
2. Create appropriate event handlers for:

- Participant join events
- Quiz state changes
- Connection status updates

Security Considerations

1. Implement basic validation to prevent:
   - Empty or excessively long participant names
   - SQL injection or XSS attacks
   - Multiple participants with identical names
2. Handle session management securely:
   - Store participant session information securely
   - Prevent unauthorized quiz access

Technical Implementation Notes

1. For QR code generation:
   - Use a reliable PHP QR code library
   - Generate codes at an appropriate size for classroom display
   - Ensure proper contrast for easy scanning
2. For the real-time participant list:
   - Implement clean animations for new participants
   - Limit the list height with scrolling for large classes
   - Consider showing a count of total participants
3. For the subdirectory installation:
   - Ensure all URLs correctly account for the `/zapquiz` prefix
   - Update URL generation in QR codes accordingly

Deliverables

The completed implementation should include:

1. Teacher dashboard with Start Quiz functionality
2. Quiz splash page with QR code, join link, and participant list
3. Participant join flow with name entry and waiting screen
4. Real-time communication system for participant updates
5. Database schema updates for sessions and participants
6. Complete integration with existing teacher authentication system

Next Steps

After this phase is complete, the third phase will implement the actual quiz gameplay, including question display, answer submission, scoring, and results visualization.

Revision #10
Created 2025-04-05 10:34:06 UTC by Max
Updated 2025-04-05 20:15:42 UTC by Max