Team Quiz

? Complete Build Prompt

Project name: Team Bug-Fix Race (ROC Amstelland Edition)

Stack: PHP 8.1, MariaDB, Apache, Vanilla JS, Tailwind CSS.

Branding: ROC Amstelland – red as accent (#e11d48 or Tailwind rose-600/red-600). UI language: **English**.

1) One?paragraph summary for the builder

Build a lightweight, real-time(ish) classroom web app where student **teams** submit a single open-text answer per question. A **teacher** controls the match, creates rounds from a pre-entered question list, and **judges** answers manually. Scoring is **6/3/1** for the first/second/third *accepted* answers per question based on the server timestamp of the submissions. A team may submit **once per question**. The teacher starts/stops each question without a time limit. Live views show the current question, incoming answers for the teacher, and a public **leaderboard**. Authentication is minimal: teacher login + **team join by 5-char code**.

2) User roles & pages

Teacher

- **Login page** (email + password)
-
  - Create/manage **Matches** (a match = N questions selected from the bank)
  - Start/stop **Current Question**

- **Answers Moderation**: live list of all team submissions (sorted by server time); buttons: *Accept* / *Reject*
- **Scoreboard view** (open in beamer window)
- Question Bank CRUD (manual input via form; title, description, code snippet, tags)
  Dashboard

Student (Team)

- **Join page**: enter *Team name* and **join code** (5 alphanumeric chars). No password. Code generated by teacher when creating teams.
- **Play page**: shows current question (read-only), text field to submit one answer (disabled after submit or when question is closed). Shows submission status.

Public/Share

- **Leaderboard page**: responsive board with team rank, score, last question deltas.

3) Gameplay rules (authoritative spec)

- Teacher creates a **Match**, adds/arranges N questions (from bank) → starts the match.
- - Teacher clicks **Open Question** → teams can submit once.
  - Each submission receives a **server-side timestamp** (UTC) that decides order.
  - Teacher views all answers (team, text, timestamp) and marks them **Accepted** or **Rejected**.
  - The first 3 **Accepted** answers award **6, 3, 1** points respectively. If fewer than 3 accepted, only those points apply.
  - If **no accepted answers**, everyone gets **0**.
  - Teacher clicks **Close Question** → submissions disabled → scores lock.
    For each question:

- Proceed to next question until all are done.

Constraints:

- Each team: **max 1 submission per question**.
- Order is determined solely by **server timestamp** when submission is stored.
- Teacher can **revoke** an accept/reject before the question is closed; if acceptance order changes, points recompute.

4) Real?time options (choose one at build time)

1. **Short-polling** (default MVP): client polls JSON endpoints every 1–2s. Easiest to host on Apache/PHP.
2. **SSE (Server-Sent Events)**: simple one-way push from server to clients for live updates (scoreboard/answers).
3. **WebSocket (Ratchet/Swoole)**: lowest latency, most complex. Optional v2.

> **Implement MVP with short-polling**, keep code structured so SSE can be swapped in later.

5) Data model (MariaDB)

```
-- users (teachers only)
CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  email VARCHAR(190) UNIQUE NOT NULL,
  password_hash VARCHAR(255) NOT NULL,
  created_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB;

-- question bank
```

```
CREATE TABLE questions (
  id INT AUTO_INCREMENT PRIMARY KEY,
  title VARCHAR(255) NOT NULL,
  body TEXT NOT NULL,             -- markdown/text
  code_snippet TEXT NULL,          -- optional fenced code
  tags VARCHAR(255) NULL,
  created_by INT NOT NULL,
  created_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (created_by) REFERENCES users(id)
) ENGINE=InnoDB;

-- matches (a session for a class)
CREATE TABLE matches (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  status ENUM('draft','active','finished') NOT NULL DEFAULT 'draft',
  created_by INT NOT NULL,
  created_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (created_by) REFERENCES users(id)
) ENGINE=InnoDB;

-- match_questions (ordered list)
CREATE TABLE match_questions (
  id INT AUTO_INCREMENT PRIMARY KEY,
  match_id INT NOT NULL,
  question_id INT NOT NULL,
  position INT NOT NULL,                      -- 1..N order
  status ENUM('pending','open','closed') NOT NULL DEFAULT 'pending',
  opened_at DATETIME NULL,
  closed_at DATETIME NULL,
  FOREIGN KEY (match_id) REFERENCES matches(id),
  FOREIGN KEY (question_id) REFERENCES questions(id),
  UNIQUE KEY uq_match_pos (match_id, position)
) ENGINE=InnoDB;

-- teams
CREATE TABLE teams (
  id INT AUTO_INCREMENT PRIMARY KEY,
  match_id INT NOT NULL,
```

```
  name VARCHAR(120) NOT NULL,
  join_code CHAR(5) NOT NULL,                 -- generated before start
  total_score INT NOT NULL DEFAULT 0,
  created_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  UNIQUE KEY uq_match_team_name (match_id, name),
  UNIQUE KEY uq_join_code (join_code),
  FOREIGN KEY (match_id) REFERENCES matches(id)
) ENGINE=InnoDB;

-- submissions (one per team per question)
CREATE TABLE submissions (
  id INT AUTO_INCREMENT PRIMARY KEY,
  match_question_id INT NOT NULL,
  team_id INT NOT NULL,
  answer_text TEXT NOT NULL,
  created_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  verdict ENUM('pending','accepted','rejected') NOT NULL DEFAULT 'pending',
  verdict_by INT NULL,                         -- teacher user id
  verdict_at DATETIME NULL,
  FOREIGN KEY (match_question_id) REFERENCES match_questions(id),
  FOREIGN KEY (team_id) REFERENCES teams(id),
  FOREIGN KEY (verdict_by) REFERENCES users(id),
  UNIQUE KEY uq_team_once (match_question_id, team_id),
  KEY idx_mq_created (match_question_id, created_at)
) ENGINE=InnoDB;

-- per-question score ledger (derived but stored for performance)
CREATE TABLE scores (
  id INT AUTO_INCREMENT PRIMARY KEY,
  match_question_id INT NOT NULL,
  team_id INT NOT NULL,
  points INT NOT NULL,
  rank INT NOT NULL,                           -- 1,2,3 for accepted order
  created_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (match_question_id) REFERENCES match_questions(id),
  FOREIGN KEY (team_id) REFERENCES teams(id),
  UNIQUE KEY uq_score_once (match_question_id, team_id)
) ENGINE=InnoDB;
```

6) Scoring logic (deterministic)

- Compute accepted submissions for the current question ordered by created_at ASC.
- Award points by order: index 1 → 6 pts, 2 → 3 pts, 3 → 1 pt. Others: 0.
- Recompute ledger whenever teacher changes a verdict **until** the question is closed. On close, persist scores rows and update teams.total_score.
- If a previously accepted answer is revoked while **open**, recalc order and points.

Pseudocode:

```
$accepted = SELECT * FROM submissions
  WHERE match_question_id=? AND verdict='accepted'
  ORDER BY created_at ASC;
$pointsMap = [6,3,1];
for each team in accepted (i=0..2) assign $pointsMap[$i]; others 0.
```

7) HTTP routes (MVP)

Auth

- POST /auth/login (teacher) → session
- POST /auth/logout

Teacher – Matches & Questions

- GET /dashboard (HTML)
- POST /matches create
- POST /matches/:id/activate → status=active
- POST /matches/:id/finish → status=finished
- GET /matches/:id (manage view)
- POST /matches/:id/teams (bulk create with names; server generates join codes)
- POST /matches/:id/questions (attach question_id + position)

- POST /match_questions/:id/open → set status=open, opened_at=NOW(), clear pending scores
- POST /match_questions/:id/close → status=closed, persist scores → update teams.total_score

Teacher – Moderation & Live

- GET /api/match_questions/:id/submissions (JSON, order by created_at)
- POST /api/submissions/:id/verdict {accepted|rejected}
- GET /api/matches/:id/leaderboard (JSON)

Question Bank

- GET /questions (HTML list)
- GET /questions/new, POST /questions
- GET /questions/:id/edit, POST /questions/:id

Student – Join & Play

- GET /join (HTML)
- POST /join (join_code + team_name optional lock)
- GET /play (HTML)
- GET /api/current (JSON: current match_question, sanitized question fields, status)
- POST /api/submit (team session must exist; if already submitted for this question → 409)
- GET /api/my-status (has_submitted, verdict if any)

Public Leaderboard

- GET /leaderboard/:matchId (HTML)
- `GET /api/leaderboard

Team

Revision #1
Created 2025-10-03 20:24:35 UTC by Max
Updated 2025-10-03 20:27:27 UTC by Max