**Build a Modern Web-Based Training & Exam Platform**

**Technology Stack:**

* **Frontend:** HTML5, CSS3, Bootstrap 5, Vanilla JavaScript (ES6+)
* **Backend:** PHP 8+ (object-oriented, MVC style)
* **Database:** MySQL/MariaDB

**1. User Authentication & Accounts**

* Registration & Login: collect username, email, password.
* Secure password storage (bcrypt/argon2), input validation + CSRF tokens.
* Session management best practices (secure, HTTP-only cookies, session timeout).
* “Forgot password” flow with secure email reset link.

**2. Bilingual Question Display**

* Store each question with both German and English text in the DB.
* On page load default to German.
* Implement a language-toggle button (AJAX) that switches all visible question text to English (no page reload).

**3. Question Entity**

* Fields:
  + id (PK),
  + text\_de, text\_en,
  + optional image\_path,
  + four answer choices (choice1\_de, choice1\_en, …),
  + correct\_choice\_index (1–4).

**4. User Progress Tracking**

* Table user\_progress linking user\_id → question\_id, storing:
  + attempt\_count, correct\_count, incorrect\_count.
* In all “training” screens show for each question: green ✓ count, red ✕ count.

**5. Training Mode**

* Sequential flow through all questions (or a specified subset).
* Immediately after selection:
  + If correct: highlight choice in green.
  + If incorrect: highlight selected choice in red, also highlight correct choice in green.
* Update user\_progress.
* “Next” and “Previous” navigation.

**6. Exam Mode**

* Randomly select 30 questions per exam instance.
* 30-minute countdown timer, visible at top of page.
* No per-question feedback during exam.
* Navigation: Next/Previous arrows.
* Display “Questions answered / 30” alongside timer.
* Pause/Resume timer (with security in mind—no cheating).
* On “Submit”: evaluate all answers, then show:
  + Green highlight for correct, red for wrong, and user’s selected choice.
  + Final score /30.
* Store exam instance in exam\_history:
  + exam\_id, user\_id, timestamp, score.
* Store per-question user answers in user\_answers:
  + exam\_id, question\_id, selected\_choice.

**7. Analytics Dashboard (per user)**

* **Training Overview:**
  + Pie chart: correct vs. incorrect vs. unattempted question counts.
  + Use Chart.js (or similar) integrated in dashboard page.
* **Exam History Table:**
  + Columns: Exam #, Date/Time, Score, View Details (link).
* **Performance Trend:**
  + Line chart of exam scores over time.

**8. Admin Interface**

* CRUD for questions (German/English text, image upload).
* Secure admin login + role-based access.

**9. Responsive Design & UX**

* Mobile-first using Bootstrap grid & utilities.
* Clean, modern card layouts for questions and dashboard widgets.
* Dark/light theme toggle (persist preference via localStorage).

**10. AJAX & UX Enhancements**

* Language switch, question navigation, and progress updates via AJAX fetch calls.
* Loading spinners or skeleton loaders for async operations.

**11. Database Schema Outline**

sql

CopyEdit

-- users table

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(50) UNIQUE NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

password\_hash VARCHAR(255) NOT NULL,

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP

);

-- questions table

CREATE TABLE questions (

id INT AUTO\_INCREMENT PRIMARY KEY,

text\_de TEXT NOT NULL,

text\_en TEXT NOT NULL,

image\_path VARCHAR(255),

choice1\_de VARCHAR(255), choice1\_en VARCHAR(255),

choice2\_de VARCHAR(255), choice2\_en VARCHAR(255),

choice3\_de VARCHAR(255), choice3\_en VARCHAR(255),

choice4\_de VARCHAR(255), choice4\_en VARCHAR(255),

correct\_choice\_index TINYINT(1) NOT NULL

);

-- user\_progress

CREATE TABLE user\_progress (

user\_id INT,

question\_id INT,

attempt\_count INT DEFAULT 0,

correct\_count INT DEFAULT 0,

incorrect\_count INT DEFAULT 0,

PRIMARY KEY(user\_id, question\_id),

FOREIGN KEY (user\_id) REFERENCES users(id),

FOREIGN KEY (question\_id) REFERENCES questions(id)

);

-- exam\_history

CREATE TABLE exam\_history (

exam\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

timestamp DATETIME DEFAULT CURRENT\_TIMESTAMP,

score TINYINT NOT NULL,

FOREIGN KEY (user\_id) REFERENCES users(id)

);

-- user\_answers

CREATE TABLE user\_answers (

exam\_id INT,

question\_id INT,

selected\_choice\_index TINYINT(1),

PRIMARY KEY (exam\_id, question\_id),

FOREIGN KEY (exam\_id) REFERENCES exam\_history(exam\_id),

FOREIGN KEY (question\_id) REFERENCES questions(id)

);

**12. Security Best Practices**

* Use prepared statements / parameterized queries (PDO).
* Escape all output (HTML-encode) to prevent XSS.
* CSRF tokens on all state-changing forms.
* Rate-limit login attempts, brute-force protection.
* Validate & sanitize file uploads (images only, max size).

**13. Optional Enhancements**

* Export exam results to PDF (TCPDF or Dompdf).
* Social login (OAuth2) for Google/LinkedIn.
* Two-factor authentication (email or TOTP).
* Drag-and-drop image upload in admin panel.

**Deliverables**

1. Full implementation plan with milestones.
2. Directory & file structure outline (frontend, backend, assets).
3. Full implementation codes, html, js, php, css and sql for key all components including auth, quiz engine, AJAX, admin for training, exam, dashboard, and admin pages.
4. Deployment instructions (LAMP stack, .htaccess, SSL).
5. Security audit checklist.

please give me complete codes. avoid using comments like // Render admin questions view and // ... process all fields ...