The World Around Us 🎮🌍

A PHP + WAMP Quiz Website (Animals + Environment)

What we built

- Kids-friendly quiz game
- Multiple quizzes per game session
- Points per quiz + overall session points
- Cumulative leaderboard stored in a text file

Learning goals

By the end, you should be able to:

- Build multi-page PHP websites using **GET/POST**
- Use **sessions** to keep state (nickname, game points)
- Load questions from **text files**
- Grade quizzes + calculate scores
- Persist cumulative scores in a **JSON text file**
- Display and sort a leaderboard

Requirements recap (what the site must do)

- Ask for **nickname** at the beginning
- User chooses a quiz from 2 topics:
  - **Animals**: type the animal name
  - **Environment**: True/False facts
- Each quiz shows **4 random questions**
- Show:
  - correct / incorrect counts
  - quiz points = (correct × 2) − (incorrect × 1)
  - overall points for current game session
- Options after quiz: **new quiz**, **leaderboard**, **exit**
- Leaderboard: cumulative points across all games per nickname

Our approach (high-level design)

We split the site into pages:

1. `index.php` → enter nickname
2. `menu.php` → choose topic / leaderboard / exit
3. `quiz.php` → show 4 random questions
4. `submit_quiz.php` → grade + update scores
5. `result.php` → show quiz result + next actions
6. `leaderboard.php` → show sorted cumulative scores
7. `exit.php` → show total across all games
8. `restart.php` → clear session and restart

We share reusable code in:

- `includes/lib.php`

Folder structure (WAMP)

Put this folder in:

`C:\wamp64\www\world-around-us\`

```
world-around-us/
index.php
menu.php
quiz.php
submit_quiz.php
result.php
leaderboard.php
exit.php
restart.php
includes/
lib.php
data/
animals.txt
environment.txt
leaderboard.json
```

Open in browser:

`http://localhost/world-around-us/`

Data storage (text files)

✅ Requirement: store data in text files

We use:

- `animals.txt` and `environment.txt` for questions
- `leaderboard.json` for cumulative scores

Why JSON for leaderboard?

- Still plain text
- Easy key-value mapping: `"nickname": score`

Questions format: Animals

File: `data/animals.txt`

Each line:

```
answer|description
```

Example:

```
elephant|A large grey mammal with a trunk, lives in Africa and Asia.
penguin|A flightless bird that swims, commonly found in the Southern Hemisphere.
```

Game behavior:

- user types an answer
- wrong OR empty → counts as incorrect

Questions format: Environment

File: `data/environment.txt`

Each line:

```
T|fact
F|fact
```

Example:

```
T|Plants produce oxygen through photosynthesis.
F|The Amazon rainforest is located in Africa.
```

Game behavior:

- user selects True/False
- no selection → incorrect

PHP basics: how a PHP page works

A `.php` file can contain HTML + PHP.

```
<?php
  $name = "Tong";
?>
<h1>Hello <?= $name ?></h1>
```

- PHP runs on the server
- Output becomes HTML sent to the browser

PHP basics: GET vs POST

GET

- data in URL query string
- good for navigation and filters
- example: `quiz.php?topic=animals`

POST

- data in request body
- used for forms (login, submit quiz)

In PHP:

- `$_GET['topic']`
- `$_POST['nickname']`

PHP basics: redirect

After processing, we often redirect:

```php
header("Location: menu.php");
exit;
```

Why?

- prevents resubmitting POST on refresh
- clean flow between pages

Sessions: storing “current game” state

We use sessions to remember:

- nickname
- current game points (session total)
- current quiz questions (to prevent cheating)

Key code:

```
session_start();
$_SESSION['nickname'] = $nick;
$_SESSION['session_points'] = 0;
```

Security note: escaping HTML output

Always escape user data before showing it:

```
htmlspecialchars($s, ENT_QUOTES, 'UTF-8');
```

We wrapped it in a helper:

```
function h(string $s): string { ... }
```

Use:

```
<?= h($_SESSION['nickname']) ?>
```

Shared helpers in `includes/lib.php`

We centralize:

- input cleaning (`clean_nickname`)
- loading question files (`load_questions`)
- random selection (`pick_random_questions`)
- normalize answers (`normalize`)
- leaderboard load/save/update

Benefits:

- less duplicated code
- easier to debug + extend

Game start: `index.php`

Responsibilities:

- show form for nickname
- validate nickname (not empty)
- start session game points at 0
- redirect to menu

Core flow:

1. `POST nickname`
2. validate
3. set session values
4. redirect

Menu: `menu.php`

Responsibilities:

- show current nickname + session points
- provide links to:
    - Animals quiz
    - Environment quiz
    - Leaderboard
    - Exit

Links:

- `quiz.php?topic=animals`
- `quiz.php?topic=environment`

Quiz screen: `quiz.php`

Responsibilities:

- read `topic` from GET
- load questions from correct text file
- randomly pick 4 questions
- store current quiz in session

Important:
✅ requirement: 4 random questions per quiz

We store quiz items in:

```
$_SESSION['current_quiz']
```

Rendering questions (Animals vs Environment)

Animals:

- show description
- input text field

Environment:

- show fact
- radio: True / False

We switch by:

```
if ($topic === 'animals') { ... } else { ... }
```

Submit quiz: `submit_quiz.php`

Responsibilities:

- read `$_SESSION['current_quiz']`
- read user answers from POST
- compute:
    - correct count
    - incorrect count
    - quiz points

Points formula:
(correct × 2) − (incorrect × 1)

Then:

- update session total
- update leaderboard file
- redirect to results

Grading logic

For each question:

- Animals: correct only if typed answer matches (empty → incorrect)
- Environment: correct only if T/F selected and matches (empty → incorrect)

We normalize strings:

- trim
- lowercase
- collapse whitespace

```
normalize("  Polar   Bear ") → "polar bear"
```

Results: `result.php`

Responsibilities:

- display last quiz result:
  - correct / incorrect
  - points from quiz
- display overall session points
- navigation options:
  - new quiz (animals/environment)
  - leaderboard
  - exit

This matches the “after quiz” requirement.

Leaderboard: `leaderboard.php`

Stored in `data/leaderboard.json` like:

```
{
  "Alice": 7,
  "Bob": 2
}
```

Page responsibilities:

- load data
- show table nickname + points
- allow sorting:
    - by nickname
    - by greatest score

Sorting in PHP (concept)

We build an array of rows:

```
$rows[] = ['name' => $name, 'score' => (int)$score];
```

Then sort:

- by score:

```
usort($rows, fn($a,$b) => $b['score'] <=> $a['score']);
```

- by name:

```
usort($rows, fn($a,$b) => strcmp(strtolower($a['name']), strtolower($b['name'])));
```

File I/O: leaderboard save safely

Important concept: **file locking** to avoid corruption if two users update.

Pattern:

- open file
- `flock(LOCK_EX)`
- overwrite JSON
- unlock + close

This makes scoreboard updates more reliable.

Exit: `exit.php`

Requirement:

- show nickname
- show **overall points across all games** (previous + current)

Because we update leaderboard every quiz,
we can read total from `leaderboard.json` and show it.

Also provide “Start new game” link.

Restart: `restart.php`

Clears session and goes back to nickname entry:

```php
session_start();
session_unset();
session_destroy();
header("Location: index.php");
exit;
```

Testing checklist (demo plan)

✅ Start game:

- enter nickname
- go to menu

✅ Quiz:

- animals: try correct, incorrect, empty
- environment: select T/F and also try leaving blank

✅ Results:

- verify points formula
- verify session total increases across quizzes

✅ Leaderboard:

- verify score persists after restarting session
- verify sorting works

✅ Exit:

- shows cumulative score

Common mistakes & fixes

- Forgot `session_start()` → session values “disappear”
- Using `$_GET` when form uses POST (or vice versa)
- Not validating `topic` → invalid file access
- Not escaping nickname output → XSS risk
- Not storing quiz in session → users can change answers/questions

Summary

You implemented a complete PHP web app using:

- multi-page navigation (GET links + POST forms)
- sessions for game state
- text files for questions + leaderboard
- random question generation
- scoring + persistent cumulative leaderboard

Next steps (optional ideas):

- add CSS
- show correct answers after quiz
- add difficulty levels
- track number of games played