Adding Role-Based Access Control

Role-based access control allows authenticated users to perform only the actions permitted by their stored role. This example uses two roles: `user` and `admin`.

Database role

A suitable users table contains:

```
role VARCHAR(20) NOT NULL DEFAULT 'user'
```

Ordinary registration must always create `user` accounts. Promote a fictional test account through a controlled administrator process or directly in the local development database:

```
UPDATE users
SET role = 'admin'
WHERE username = 'admin_demo';
```

Do not allow a public form to submit its own role.

Store the trusted role at login

After verifying the password:

```
session_regenerate_id(true);
$_SESSION["user_id"] = (int) $user["user_id"];
$_SESSION["username"] = $user["username"];
$_SESSION["role"] = $user["role"];
```

Require an administrator

```php
<?php
session_start();

if (!isset($_SESSION["user_id"])) {
    header("Location: login.php");
    exit;
}

if (($_SESSION["role"] ?? "") !== "admin") {
    http_response_code(403);
    exit("You do not have permission to access this page.");
}
```

Authentication asks “Who is signed in?” Authorisation asks “May that user perform this action?” Both checks are required.

Reusable administrator guard

Create `includes/require-admin.php` containing the checks above, then require it from every administrative display and processing route.

Test matrix

Account state	User page	Admin page
Logged out	Redirect to login	Redirect to login
Standard user	Allowed	403 response
Administrator	Allowed	Allowed
Changed form/URL value	No role change	No additional access

Check

- [ ] Roles are `user` and `admin`.
- [ ] Role comes from the stored user record.
- [ ] Administrative processing is protected.
- [ ] Direct URLs are tested.

- [ ] Denied access uses an appropriate response.

Revision #2
Created 2026-06-08 08:56:10 UTC by Mr Napper
Updated 2026-08-18 23:49:58 UTC by Mr Napper