

Done adding url. Let's solve conflicting URLs issue first. What you prefer: (1) a function that checks if the url exists any of these tables (2) a url_registery table. Which approach is better in terms of minimal and clean setup? Given your modular architecture and preference for clarity, I'd recommend **Option 1: a function that checks across all slug-enabled tables**. It's minimal, clean, and aligns with your current routing strategy—especially since you're caching pages individually and not relying on dynamic lookups for every request.

✓ Why Option 1 Is Better for You

Criteria	Function-Based Lookup	url_registery Table
🔧 Minimal setup	No new tables or sync logic	Requires syncing slugs manually
🗑️ Editorial clarity	Slugs live in their native tables	Adds indirection for editors
⚡ Performance (cached)	Fast due to static page caching	Slightly faster for dynamic sites
🔗 Maintainability	Easy to extend with new modules	Needs registry updates per insert

Since you've already added url fields to all relevant tables, we can now build a **single PHP function** that checks for slug existence across all modules before insert/update.

🔗 Suggested Function: isUrlTaken(\$slug)

```
php
function isUrlTaken($slug, $excludeTable = '', $excludeKey = 0) {
    global $conn;

    $tables = [
        'articles' => 'key_articles',
        'pages' => 'key_pages',
        'categories' => 'key_categories',
        'books' => 'key_books',
        'products' => 'key_product',
        'authors' => 'key_authors',
        'photo_gallery' => 'key_photo_gallery',
        'youtube_gallery' => 'key_youtube_gallery',
        'blocks' => 'key_blocks',
        'main_menu' => 'key_main_menu',
        'book_categories' => 'key_book_categories',
        'product_categories' => 'key_product_categories',
        'photo_categories' => 'key_photo_categories',
        'youtube_categories' => 'key_youtube_categories',
        'users' => 'key_user'
    ];

    foreach ($tables as $table => $keyField) {
        if ($table === $excludeTable) continue;

        $stmt = $conn->prepare("SELECT COUNT(*) FROM $table WHERE url = ?" . ($excludeKey ? "
AND $keyField != ?" : ""));
        if ($excludeKey) {
            $stmt->bind_param("si", $slug, $excludeKey);
        } else {
            $stmt->bind_param("s", $slug);
        }

        $stmt->execute();
        $stmt->bind_result($count);
        $stmt->fetch();
        $stmt->close();

        if ($count > 0) return true;
    }

    return false;
}
```

✂ Usage Example

php

```
if (isUrlTaken($_POST['url'], 'products', $key_product)) {
    echo "<p>✂ This URL is already used in another module. Please choose a unique one.</p>";
} else {
    // Proceed with insert/update
}
```

Let me know if you want to auto-suggest a unique slug (e.g. `my-title-2`) or build a dashboard widget to audit all current slugs for conflicts. We can even log duplicates for editorial review.

Here is the `add.php`, it does show the message that the url exists in another module, but how do I send message back to the modal that submitted the form:

```
<?php include '../db.php'; include '../users/auth.php'; if ($_SERVER['REQUEST_METHOD'] === 'POST') { if
(isUrlTaken($_POST['url'], 'products', $key_product)) { echo "<p>✂ This URL is already used in another
module. Please choose a unique one.</p>"; exit; } else { $status = isset($_POST['status']) ? 'on' : 'off'; $createdBy
= $_SESSION['key_user']; $stmt = $conn->prepare("INSERT INTO photo_gallery ( title, image_url, description,
status, created_by ) VALUES (?, ?, ?, ?, ?)"); if (!$stmt) { die("Prepare failed: " . $conn->error); } $stmt-
>bind_param("ssssi", $_POST['title'], $_POST['image_url'], $_POST['description'], $status, $createdBy ); $stmt-
>execute(); $newRecordId = $conn->insert_id; if (!empty($_POST['categories'])) { $stmtCat = $conn-
>prepare("INSERT IGNORE INTO photo_categories (key_photo_gallery, key_categories) VALUES (?, ?)");
foreach ($_POST['categories'] as $catId) { $stmtCat->bind_param("ii", $newRecordId, $catId); $stmtCat-
>execute(); } } } header("Location: list.php"); exit; Here is the modal that submits the form: <!-- Modal Form
--> <div id="modal" class="modal"> <h3 id="modal-title">Add Photo</h3> <form id="modal-form"
method="post" action="add.php"> <input type="hidden" name="key_photo_gallery" id="key_photo_gallery">
<input type="text" name="title" id="title" placeholder="Title" required><br> <input type="text"
name="image_url" id="image_url" placeholder="Image URL" required><br> <textarea name="description"
id="description" placeholder="Description"></textarea><br> <label> <input type="checkbox" name="status"
id="status" value="on" checked> Active </label><br> <div id="select-categories"> <h3>Categories</h3> <?php
$types = ['photo_gallery', 'book', 'article', 'video_gallery', 'global']; foreach ($types as $type) { echo "<div
style='color:margin:10px 0;*>"; echo "<div style='color:Navy;padding:10px 0 10px 0;*>" . ucfirst(str_replace('_', '
', $type)) . "</div>"; $catResult = $conn->query("SELECT key_categories, name FROM categories WHERE
category_type = '$type' AND status='on' ORDER BY sort"); while ($cat = $catResult->fetch_assoc()) { echo
"<label style='display:block;*> <input type='checkbox' name='categories[]' value='{ $cat['key_categories']}'>
{ $cat['name']} </label>"; } echo "</div>"; } ?> </div> <input type="submit" value="Save"> <button
type="button" onclick="closeModal()">Cancel</button> </form> </div>
```

Great question, Nasiruddin bhai. To send the error message back to the modal without redirecting or reloading the page, you'll want to switch from a standard form submission to an **AJAX-based submission**. This way, the PHP script can return a response (like a JSON object or plain text), and your frontend can display it inside the modal dynamically.

✓ Step-by-Step Solution

1. Update the Form to Use JavaScript

Add an `id` to your form (already done: `modal-form`) and prevent default submission:

html

```
<script>
document.getElementById('modal-form').addEventListener('submit', function(e) {
    e.preventDefault();

    const form = e.target;
    const formData = new FormData(form);

    fetch(form.action, {
        method: 'POST',
        body: formData
    })
    .then(response => response.text())
```

```

.then(data => {
  if (data.includes('✖')) {
    document.getElementById('modal-title').innerHTML = data;
  } else {
    window.location.href = 'list.php'; // or close modal and refresh list
  }
})
.catch(error => {
  document.getElementById('modal-title').innerHTML = '✖ Submission failed.';
  console.error(error);
});
});
</script>

```

2. Update add . php to Return Plain Text

Instead of redirecting immediately, return a success message or error:

php

```

if ($_SERVER['REQUEST_METHOD'] === 'POST') {

    if (isUrlTaken($_POST['url'], 'photo_gallery')) {
        echo "✖ This URL is already used in another module. Please choose a unique one.";
        exit;
    }

    // ... your insert logic ...

    echo "✓ Photo added successfully.";
    exit;
}

```

Note: I changed 'products' to 'photo_gallery' in `isUrlTaken()` since this is the photo module.

3. Optional: Show Message in a Dedicated Area

Instead of replacing the modal title, you can add a `<div id="modal-message"></div>` below the form and update that:

js

```
document.getElementById('modal-message').innerHTML = data;
```

Let me know if you want to auto-close the modal after success or refresh the list dynamically. We can even animate the message for better UX.

I'm going to update php files to add and edit url values. Then I will work on the `isUrlTaken` function after carefully reviewing the logic.

That sounds like a solid recovery plan, Nasiruddin bhai.

Once you've updated the PHP files to handle url values properly in both add and edit flows, we can revisit `isUrlTaken()` with fresh eyes and make sure it's airtight. I'll be here to help you:

- ✓ Validate the logic across modules
- 🔄 Handle edge cases like self-matching during updates
- ⚡ Optimize performance if needed (e.g. early exit, caching)
- 🎯 Add meaningful error messages for editors

Whenever you're ready to dive back in, just drop the updated code or schema and we'll pick up right where you left off—with clarity and care.

Today

Alright, what I did is attached the submit listener to all the add/edit modal forms and removed their action attributes. Now forms are being submitted using the listener, not the action attribute. I will work on the

intercepting message in case of errors, but right now I need a JS function that cleans up a url (slug). I wanna use this function on title input element as onchange="setCleanURL(this.value);" What you think? That’s a clean and scalable move, Nasiruddin bhai—centralizing form submission via JS gives you full control over validation, error handling, and dynamic feedback. Now for the slug cleaner: I’ve got just the function you need.

✓ JavaScript Slug Cleaner: setCleanURL(title)

```
js
function setCleanURL(title) {
  const slug = title
    .toLowerCase()
    .trim()
    .replace(/[^\a-z0-9\s-]/g, '')      // Remove special chars
    .replace(/\s+/g, '-')              // Replace spaces with hyphens
    .replace(/-+/g, '-')               // Collapse multiple hyphens
    .replace(/^|-$/g, '');             // Trim leading/trailing hyphens

  document.querySelector('[name="url"]').value = slug;
}
```

How It Works

- Converts to lowercase
- Removes punctuation and symbols
- Replaces spaces with hyphens
- Collapses multiple hyphens
- Trims edges

Usage in HTML

```
html
<input type="text" name="title" onchange="setCleanURL(this.value)" placeholder="Title">
<input type="text" name="url" placeholder="Slug">
```

You can also call setCleanURL() on keyup or blur if you want real-time updates or delayed cleanup.

Let me know if you want to auto-check for duplicates after generating the slug, or show a warning if the slug already exists. We can even color the input red or show a tooltip. Your frontend is shaping up beautifully.

The thing is when the title of an article, book, product, category is updated or changed, it should not then change the url, the url might already been published. Exactly, Nasiruddin bhai—that’s a crucial editorial safeguard. Once a ur l (slug) is published and indexed (by search engines, shared links, or internal references), it becomes part of your site's identity. Changing it casually can:

- ✖ Break external links
- ✖ Hurt SEO rankings
- ✖ Confuse returning users or editors

✓ Best Practice: Decouple Title and URL

Here’s how to handle it cleanly:

1. Generate URL only on initial creation

Use your setCleanURL(title) function only when the form is in “Add” mode—not during edits.

```
js
function setCleanURL(title) {
```

```
const urlInput = document.querySelector('[name="url"]');
if (!urlInput.value) {
  const slug = title
    .toLowerCase()
    .trim()
    .replace(/^[a-z0-9\s-]/g, '')
    .replace(/\s+/g, '-')
    .replace(/-+/g, '-')
    .replace(/^-|-$/g, '');
  urlInput.value = slug;
}
}
```

This way, if the `url` field already has a value (from an existing record), it won't be overwritten.

2. Make URL field editable but cautious

Let editors manually change the slug if needed, but show a warning:

html

```
<input type="text" name="url" placeholder="Slug">
<small style="color:gray;">Changing the URL may break existing links.</small>
```

3. Log URL changes (optional)

If you want to track slug changes for redirects or legacy support:

- Add a `url_history` table
- Log old slugs when a record is updated
- Use it to create 301 redirects or show a “This page has moved” notice

Let me know if you want to build a redirect handler or a slug audit tool that flags risky changes. Your editorial workflow is already thoughtful—this will make it bulletproof.

Alright this part is done, I will look into the `isUrlTaken` function later. We need a single place for message display, where no scroll up or down is needed to find out if there is a message generated. This place can be at the top of the page with a dedicated div, which appears for a while and disappears, or it has an 'x' icon to close it manually. But we do it later, take care bye. Thanks a lot for your tech help.

You're most welcome, Nasiruddin bhai. You've made thoughtful, strategic progress today—and that message display idea is spot-on for editorial clarity and UX polish. When you're ready to implement it, we'll make it elegant, reusable, and unobtrusive.

Until then, take care of yourself and your craft. Your CMS is becoming a legacy worth honoring. Catch you next time, Cobuddy.