PHP: remote website fetcher

index.php

```php
<?php
function fetchPage($url)
{
    // Parse the URL to get base components
    $parsed_url = parse_url($url);
    $base_url = $parsed_url['scheme'] . '://' . $parsed_url['host'];
    if (isset($parsed_url['port'])) {
        $base_url .= ':' . $parsed_url['port'];
    }
    $base_path = dirname($parsed_url['path'] ?? '/');
    if ($base_path !== '/') {
        $base_path .= '/';
    }

    // Use cURL to fetch the page content
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_FOLLOWLOCATION, true);
    curl_setopt($ch, CURLOPT_TIMEOUT, 10);
    curl_setopt($ch, CURLOPT_USERAGENT, 'Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36');

    $content = curl_exec($ch);
    $http_code = curl_getinfo($ch, CURLINFO_HTTP_CODE);
    curl_close($ch);

    if ($http_code !== 200 || !$content) {
        return false;
    }

    // Create a DOM parser to properly handle HTML
    $dom = new DOMDocument();
    @$dom->loadHTML('<?xml encoding="UTF-8">' . $content);

    // Fetch and inline CSS stylesheets
    $xpath = new DOMXPath($dom);
    $links = $xpath->query("//link[@rel='stylesheet']");

    $css_content = "";
    foreach ($links as $link) {
        $href = $link->getAttribute('href');
        $css_url = resolveUrl($href, $base_url, $base_path);
        $css = fetchResource($css_url);
        if ($css) {
```

```
            $css_content .= "/* From: " . htmlspecialchars($css_url) .
" */\n";
            $css_content .= $css . "\n\n";
        }
        $link->parentNode->removeChild($link);
    }

    // Also grab any inline styles in <style> tags and rewrite URLs in
them
    $styles = $xpath->query("//style");
    foreach ($styles as $style) {
        $style_text = $style->textContent;
        // Fix URLs in CSS (like background-image: url(...))
        $style_text = preg_replace_callback(
            '/url\([\'"]?(?!(?:data:|https?:|\/\/))([^\)]+)[\'"]?\)/i',
            function($matches) use ($base_url, $base_path) {
                $resource_url = resolveUrl($matches[1], $base_url,
$base_path);
                $image_data = fetchImageAsDataUri($resource_url);
                if ($image_data) {
                    return 'url(' . $image_data . ')';
                }
                return 'url(' . $resource_url . ')';
            },
            $style_text
        );
        $style->textContent = $style_text;
    }

    // Insert inlined CSS into head
    $head = $xpath->query("//head")[0];
    if ($head && !empty($css_content)) {
        $style_element = $dom->createElement('style');
$style_element->appendChild($dom->createTextNode($css_content));
        $head->insertBefore($style_element, $head->firstChild);
    }

    // Convert all images to base64 data URIs
    $images = $xpath->query("//img");
    foreach ($images as $img) {
        if ($img->hasAttribute('src')) {
            $src = $img->getAttribute('src');
            $resolved = resolveUrl($src, $base_url, $base_path);
            $image_data = fetchImageAsDataUri($resolved);
            if ($image_data) {
                $img->setAttribute('src', $image_data);
            } else {
                $img->setAttribute('src', $resolved);
            }
        }
    }
```

```
    // Fix all relative URLs for scripts, and links (but not images)
    $resources = $xpath->query("//*[@href]");
    foreach ($resources as $element) {
        if ($element->tagName !== 'link') {
            $href = $element->getAttribute('href');
            // Don't modify anchor links or javascript
            if (!preg_match('/^(#|javascript:|mailto:|tel:)/', $href))
{
                $resolved = resolveUrl($href, $base_url, $base_path);
                $element->setAttribute('href', $resolved);
            }
        }
    }

    // Extract only the body content to avoid nested html/body tags
    $body = $xpath->query("//body")[0];
    if ($body) {
        $inner_html = '';
        foreach ($body->childNodes as $node) {
            $inner_html .= $dom->saveHTML($node);
        }
        return $inner_html;
    }

    return $dom->saveHTML();
}

function resolveUrl($relative_url, $base_url, $base_path)
{
    // If it's already absolute, return as-is
    if (preg_match('~^(?:f|ht)tps?://~i', $relative_url)) {
        return $relative_url;
    }

    // If it starts with //, add the scheme from base_url
    if (preg_match('~^//~', $relative_url)) {
        $parsed = parse_url($base_url);
        return $parsed['scheme'] . ':' . $relative_url;
    }

    // If it starts with /, it's from the root
    if (preg_match('~^/~', $relative_url)) {
        return $base_url . $relative_url;
    }

    // Otherwise, it's relative to the current path
    return $base_url . $base_path . $relative_url;
}

function fetchResource($url)
```

```
{
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_FOLLOWLOCATION, true);
    curl_setopt($ch, CURLOPT_TIMEOUT, 10);
    curl_setopt($ch, CURLOPT_USERAGENT, 'Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36');

    $content = curl_exec($ch);
    $http_code = curl_getinfo($ch, CURLINFO_HTTP_CODE);
    curl_close($ch);

    return ($http_code === 200) ? $content : false;
}

function fetchImageAsDataUri($url)
{
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_FOLLOWLOCATION, true);
    curl_setopt($ch, CURLOPT_TIMEOUT, 10);
    curl_setopt($ch, CURLOPT_USERAGENT, 'Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36');

    $image_data = curl_exec($ch);
    $http_code = curl_getinfo($ch, CURLINFO_HTTP_CODE);
    $content_type = curl_getinfo($ch, CURLINFO_CONTENT_TYPE);
    curl_close($ch);

    if ($http_code !== 200 || !$image_data) {
        return false;
    }

    // Determine MIME type
    if (!$content_type) {
        $finfo = finfo_open(FILEINFO_MIME_TYPE);
        $content_type = finfo_buffer($finfo, $image_data);
        finfo_close($finfo);
    }

    // Only convert common image types
    if (!preg_match('~image/(jpeg|png|gif|webp|svg\+xml)~i',
$content_type)) {
        return false;
    }

    // Encode to base64 and create data URI
    $base64 = base64_encode($image_data);
    return 'data:' . $content_type . ';base64,' . $base64;
```

```
}

$error = null;
$content = null;

if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['url'])) {
    $url = trim($_POST['url']);

    // Validate and normalize URL
    if (!preg_match('~^https?://~i', $url)) {
        $url = 'https://' . $url;
    }

    if (filter_var($url, FILTER_VALIDATE_URL)) {
        $content = fetchPage($url);
        if ($content === false) {
            $error = "Failed to fetch the URL. The website may be
unreachable or blocked.";
        }
    } else {
        $error = "Invalid URL format.";
    }
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Website Fetcher</title>
    <style>
        * {
            box-sizing: border-box;
        }

        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 20px;
            background-color: #f5f5f5;
        }

        .container {
            max-width: 1200px;
            margin-left: auto;
            margin-right: auto;
            background-color: white;
            padding: 20px;
            border-radius: 4px;
```

```
            box-shadow: 0 2px 4px rgba(0,0,0,0.1);
        }

        input[type="text"] {
            width: 70%;
            padding: 8px;
            font-size: 14px;
            border: 1px solid #ccc;
            border-radius: 4px;
        }

        input[type="submit"] {
            padding: 8px 15px;
            cursor: pointer;
            font-size: 14px;
            background-color: #007bff;
            color: white;
            border: none;
            border-radius: 4px;
        }

        input[type="submit"]:hover {
            background-color: #0056b3;
        }

        .error {
            color: #721c24;
            padding: 10px;
            background-color: #f8d7da;
            border: 1px solid #f5c6cb;
            border-radius: 4px;
            margin: 10px 0;
        }

        .content-wrapper {
            border: 1px solid #ddd;
            padding: 20px;
            margin-top: 20px;
            border-radius: 4px;
            background-color: white;
            overflow-x: auto;
        }

        .content-wrapper * {
            max-width: 100%;
            height: auto;
        }

        .content-wrapper img {
            max-width: 100%;
            height: auto;
```

```
        }

        h1, h2 {
            color: #333;
        }

        a {
            color: #007bff;
        }

        a:hover {
            text-decoration: underline;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Website Fetcher</h1>
        <p>Fetch and view websites privately. All content is processed
server-side.</p>
        <p>Learn more: <a
href="https://wiki.ostrowski.net.pl/doku.php?id=en:narzedzia:php_websit
e_fetch" target="_blank">wiki.ostrowski.net.pl</a></p>
        <p>Yes it is wonky URL handling doesnt work,<br> CSS is loaded
only partially,<br> interactive parts of pages will not work,<br> but
that is the price of browsing privately :3.</p>
        <p>When it comes to the URLs you need to copy and paste them in
again in the search box</p>
        <form method="post">
            <input type="text" name="url" placeholder="Enter URL (e.g.,
example.com or https://example.com)" required>
            <input type="submit" value="Fetch">
        </form>

        <?php if ($error): ?>
            <div class="error"><?= htmlspecialchars($error) ?></div>
        <?php elseif ($content): ?>
            <h2>Fetched Content:</h2>
            <div class="content-wrapper">
                <?= $content ?>
            </div>
        <?php endif; ?>

        <p><small>All resource fetching happens server-side for privacy
and security.</small></p>
    </div>
</body>
</html>
```