EGOTECHWORLD
egotechworld.com · Beginner Tutorials

Simple Payment System in PHP

Step-by-Step Guide for Beginners

Build a 3-page checkout flow that delivers a downloadable PDF for $5.

What you'll build

1. index.php — card details form with a Pay button
2. processing.php — validates and processes the payment
3. payment-done.php — shows receipt and PDF download link

Skill level: Absolute beginner · Time: ~30 minutes

Requires only PHP and a basic browser. No database needed.

Introduction

In this tutorial you will build a small but complete payment system from scratch using only PHP. The goal is to take a customer through three clean pages: a card form, a processing screen, and a thank-you page where they can download a PDF file they purchased for $5.

This is a learning project. We will **simulate** the payment so you can focus on understanding how the pages connect. At the end we will show you exactly where to plug in a real payment provider when you are ready.

What you will learn

- How HTML forms send data to PHP using POST
- How to validate user input in a safe way
- How sessions remember a user between pages
- How to deliver a downloadable file after payment
- How to lay out a clean white interface using Bootstrap 5

What you need before you start

- A computer with PHP installed (XAMPP, MAMP, or LankaHost works)
- A code editor like VS Code or Notepad++
- A web browser (Chrome, Firefox, or Edge)
- A PDF file you want to sell, named **tutorial.pdf**

Note: You do not need a database for this tutorial. We use PHP sessions to pass information between pages. This keeps everything simple so you can run it on any cheap shared hosting.

Project Structure

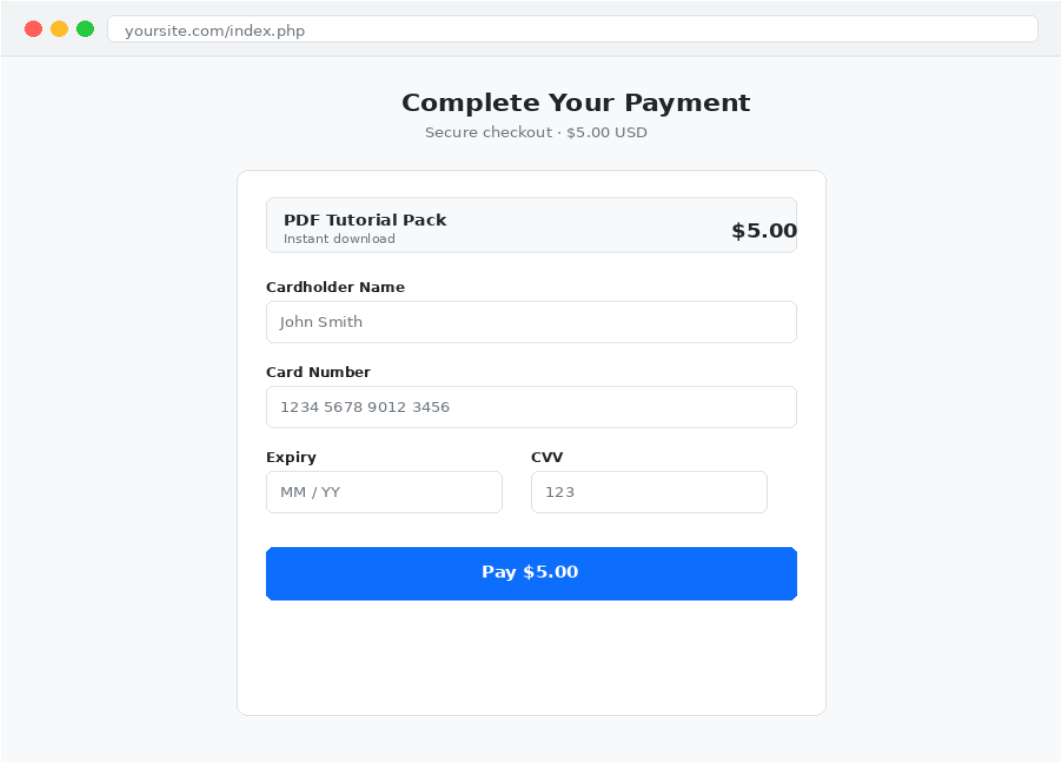
Create a new folder called **simple-payment** on your computer or on your hosting account. Inside it you will create four items in total: three PHP files and one downloads folder.

```
simple-payment/
  |
  +-- index.php           <- card details form (start here)
  +-- processing.php      <- processes the payment
  +-- payment-done.php    <- shows receipt + download link
  |
  +-- downloads/
        +-- tutorial.pdf  <- the file the customer is buying
```

Tip: Put your real PDF file inside the **downloads/** folder and rename it to **tutorial.pdf**. The download link in step 3 will look for that exact filename.

Step 1 — Build the Card Form

The first page is what the customer sees when they click "Buy". It shows the product, the price, and a simple form to enter card details. Below is what we are about to build.



Preview: index.php — clean checkout form with one Pay button

How this page works

We use a standard HTML **<form>** tag. When the customer clicks the Pay button, the browser sends all four fields (name, card number, expiry, and CVV) to **processing.php** using the POST method. POST is used because card details should never appear in the URL.

FILE: index.php

```
<?php
// index.php — card details form
session_start();
$price = 5.00;
$product = 'PDF Tutorial Pack';
?>
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Checkout — <?= $product ?></title>
  <link rel="stylesheet" href=
  "https://cdn.jsdelivr.net/npm/bootstrap@5.3.3
  /dist/css/bootstrap.min.css">
</head>
<body class="bg-light">
```

```
<div class="container py-5" style="max-width:560px;">
  <h2 class="text-center fw-bold">
    Complete Your Payment
  </h2>
  <p class="text-center text-muted">
    Secure checkout · $<?= number_format($price, 2) ?> USD
  </p>

  <div class="card shadow-sm border-0">
    <div class="card-body p-4">

      <!-- Order summary box -->
      <div class="d-flex justify-content-between
                  align-items-center bg-light p-3
                  rounded mb-4">
        <div>
          <div class="fw-bold"><?= $product ?></div>
          <small class="text-muted">Instant download</small>
        </div>
        <div class="fs-4 fw-bold">
          $<?= number_format($price, 2) ?>
        </div>
      </div>
```

```
      <!-- Payment form -->
      <form action="processing.php" method="POST">
        <div class="mb-3">
          <label class="form-label fw-semibold">
            Cardholder Name
          </label>
          <input type="text" name="name"
                 class="form-control"
                 placeholder="John Smith" required>
        </div>

        <div class="mb-3">
          <label class="form-label fw-semibold">
            Card Number
          </label>
          <input type="text" name="card"
                 class="form-control"
                 placeholder="1234 5678 9012 3456"
                 maxlength="19" required>
        </div>
```

```
        <div class="row">
          <div class="col-6 mb-3">
            <label class="form-label fw-semibold">
              Expiry
            </label>
            <input type="text" name="expiry"
                   class="form-control"
                   placeholder="MM / YY" required>
          </div>
          <div class="col-6 mb-3">
            <label class="form-label fw-semibold">
              CVV
            </label>
            <input type="text" name="cvv"
                   class="form-control"
                   placeholder="123" maxlength="4" required>
          </div>
        </div>

        <button type="submit"
          class="btn btn-primary w-100 py-2 fw-bold">
          Pay $<?= number_format($price, 2) ?>
        </button>
      </form>

    </div>
  </div>
</div>
</body>
</html>
```

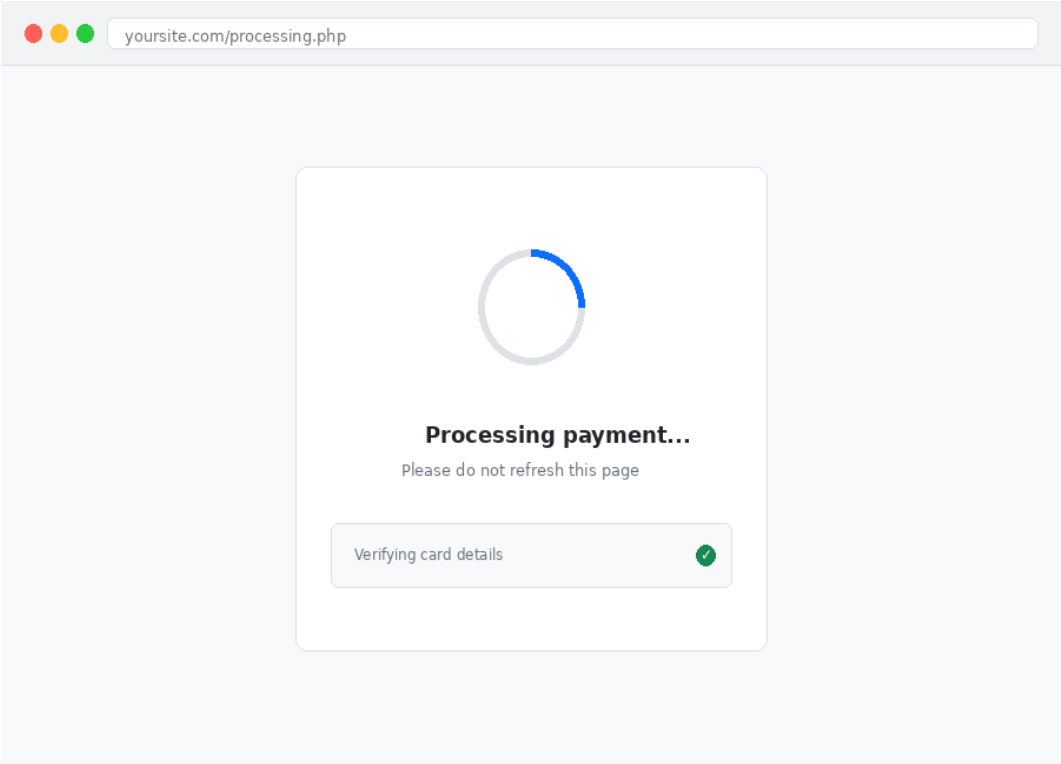
Important parts explained

- **session_start()** at the top — opens a memory area we use later.
- **action="processing.php"** — tells the form where to send data.
- **method="POST"** — sensitive data goes in the request body, not the URL.
- **required** — browsers refuse to submit if the field is empty.
- **maxlength="19"** — card numbers are 16 digits plus 3 spaces.

Tip: The **$price** variable is set in one place at the top. If you want to charge $10 later, change just that one line.

Step 2 — Process the Payment

When the customer clicks Pay, this page runs in the background. Its job is to check the data, pretend to talk to a payment company, and then send the customer to the success page. We show a small spinner so the customer knows something is happening.



Preview: processing.php — brief loading screen before redirect

How this page works

We do three things here, in order: (1) read the form fields with **$_POST**, (2) check that nothing is empty, and (3) save the result into the session so the next page can use it. After a two-second pause we redirect the browser to **payment-done.php**.

FILE processing.php

```
<?php
// processing.php — handle the payment
session_start();

// 1. Only accept POST submissions
if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
    header('Location: index.php');
    exit;
}

// 2. Read and clean the form fields
$name   = trim($_POST['name']   ?? '');
$card   = trim($_POST['card']   ?? '');
$expiry = trim($_POST['expiry'] ?? '');
$cvv    = trim($_POST['cvv']    ?? '');

// 3. Basic checks — all fields must be filled
if ($name === '' || $card === '' ||
    $expiry === '' || $cvv === '') {
    die('Please go back and fill in all fields.');
}
```

```
// 4. Card number length check (very basic)
$digits = preg_replace('/\D/', '', $card);
if (strlen($digits) < 13 || strlen($digits) > 19) {
    die('That card number does not look right.');
}

// 5. Pretend to talk to a payment company
//    In real life you would call Stripe, PayHere, etc.
$payment_success = true;   // always true in this demo

// 6. Save what we need for the next page
if ($payment_success) {
    $_SESSION['paid']     = true;
    $_SESSION['name']     = $name;
    $_SESSION['amount']   = 5.00;
    $_SESSION['order_id'] = 'ORD-' . rand(10000, 99999);
    $_SESSION['last4']    = substr($digits, -4);
}
?>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Processing your payment...</title>
  <!-- Auto-redirect after 2 seconds -->
  <meta http-equiv="refresh"
        content="2;url=payment-done.php">
  <link rel="stylesheet" href=
  "https://cdn.jsdelivr.net/npm/bootstrap@5.3.3
  /dist/css/bootstrap.min.css">
</head>
<body class="bg-light">
<div class="container py-5" style="max-width:420px;">
  <div class="card shadow-sm border-0 text-center p-5">
    <div class="spinner-border text-primary mx-auto mb-4"
         style="width:4rem; height:4rem;" role="status">
    </div>
    <h4 class="fw-bold">Processing payment...</h4>
    <p class="text-muted">Please do not refresh this page</p>
  </div>
</div>
</body>
</html>
```

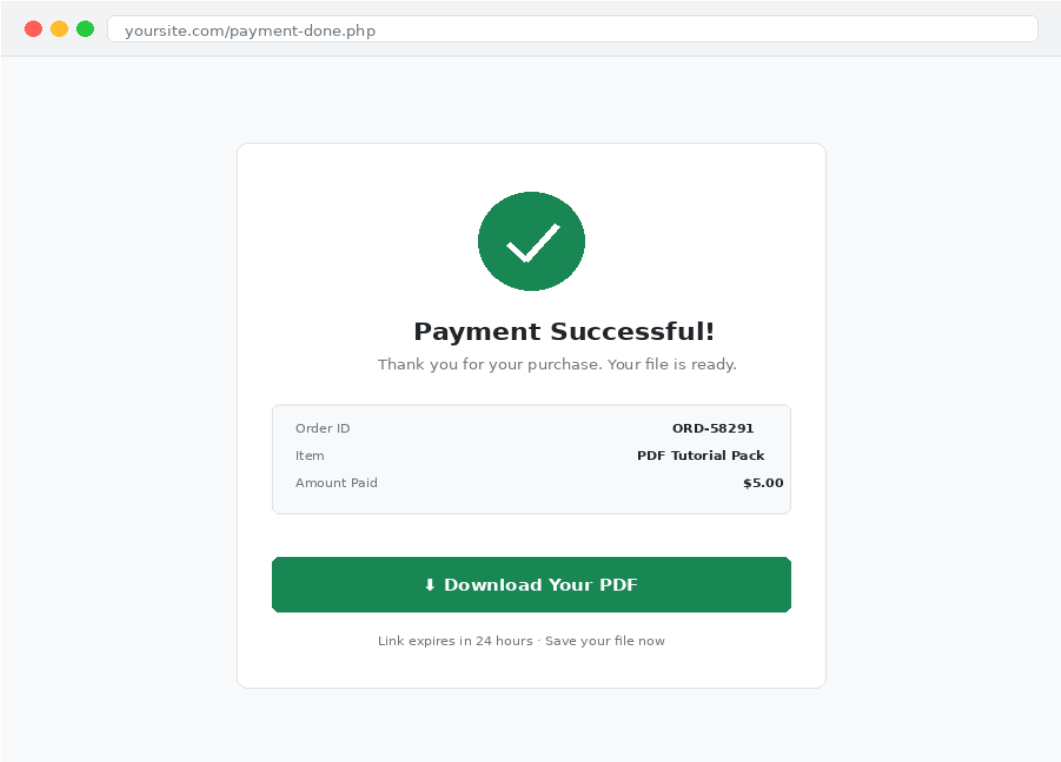
Why we use sessions here

A **session** is a small piece of memory PHP keeps for one visitor. We store the order ID and amount in the session so that the next page (**payment-done.php**) can show the receipt without us passing data through the URL. That keeps the URL clean and prevents the customer from changing the price by editing it.

Important: The line **$payment_success = true;** is a placeholder. When you switch to a real provider like Stripe or PayHere, you will replace that single line with their API call and set the variable based on their response.

Step 3 — Show Receipt and Download

This is the page customers see after a successful payment. It confirms the purchase, shows a short receipt, and gives them a button to download the PDF they paid for.



Preview: payment-done.php — receipt and download button

How this page works

Before we show anything, we check the session. If the visitor did not actually pay (for example, they typed the URL directly into the browser), we send them back to the start. Only paid visitors see the download button.

FILE payment-done.php

```
<?php
// payment-done.php — receipt and download
session_start();

// Block direct access — only paid users may see this page
if (empty($_SESSION['paid'])) {
    header('Location: index.php');
    exit;
}

$name     = $_SESSION['name']     ?? 'Customer';
$order_id = $_SESSION['order_id'] ?? 'ORD-00000';
$amount   = $_SESSION['amount']   ?? 5.00;
$last4    = $_SESSION['last4']    ?? '****';
?>
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Payment Successful</title>
  <link rel="stylesheet" href=
  "https://cdn.jsdelivr.net/npm/bootstrap@5.3.3
  /dist/css/bootstrap.min.css">
</head>
<body class="bg-light">
```

```
<div class="container py-5" style="max-width:560px;">
  <div class="card shadow-sm border-0 text-center p-4">

    <!-- Green tick -->
    <div class="mx-auto mb-3 d-flex align-items-center
                justify-content-center bg-success text-white
                rounded-circle"
         style="width:90px; height:90px; font-size:42px;">
      &check;
    </div>

    <h2 class="fw-bold">Payment Successful!</h2>
    <p class="text-muted">
      Thank you, <?= htmlspecialchars($name) ?>.
      Your file is ready to download.
    </p>
```

```
    <!-- Receipt -->
    <div class="bg-light rounded p-3 my-4 text-start">
      <div class="d-flex justify-content-between">
        <span class="text-muted">Order ID</span>
        <strong><?= $order_id ?></strong>
      </div>
      <div class="d-flex justify-content-between mt-2">
        <span class="text-muted">Item</span>
        <strong>PDF Tutorial Pack</strong>
      </div>
      <div class="d-flex justify-content-between mt-2">
        <span class="text-muted">Card</span>
        <strong>**** **** **** <?= $last4 ?></strong>
      </div>
      <div class="d-flex justify-content-between mt-2">
        <span class="text-muted">Amount Paid</span>
        <strong>$<?= number_format($amount, 2) ?></strong>
      </div>
    </div>
```

```
    <!-- Download button -->
    <a href="downloads/tutorial.pdf"
       class="btn btn-success w-100 py-2 fw-bold"
       download>
      &#x2B07; Download Your PDF
    </a>

    <small class="text-muted mt-3 d-block">
      Save your file now. This link is for one purchase.
    </small>
  </div>
</div>
</body>
</html>
```

Why we hide the card number

We only store the last four digits with **substr($digits, -4)** and display the rest as stars. Never store full card numbers on your own server — that is a job for the payment provider.

Tip: The **download** attribute on the link tells the browser to download the file instead of opening it in a new tab. It works on every modern browser.

Test Your Payment System

Now let's run it. Follow these steps in order. If something does not work, the most common causes are listed below the steps.

Run it locally

1. Place all three PHP files inside the **simple-payment** folder.
2. Create the **downloads** folder and put your **tutorial.pdf** inside.
3. If you use XAMPP, copy the folder into **htdocs**.
4. Open **http://localhost/simple-payment/index.php** in your browser.
5. Fill in any name, type 16 digits as a card number, any expiry, any CVV.
6. Click **Pay $5.00** and watch the spinner appear for 2 seconds.
7. You should land on the success page with a working download button.

Common problems

- **Page shows raw PHP code** — PHP is not running. Open the file through localhost, not by double-clicking it.
- **"Headers already sent" error** — There is a blank line or space before **<?php**. Remove it.
- **Download button does nothing** — The path to **downloads/tutorial.pdf** is wrong, or the file is missing.
- **Lands on index.php after paying** — Sessions are not enabled. Check that **session_start()** is on every page.

Going from Demo to Real Payments

This tutorial is intentionally simple. To accept real money, you need a payment provider. They handle the card details so that sensitive data never touches your server. Here is the upgrade path, from easiest to hardest.

Recommended providers

- **PayHere** — Best for Sri Lanka. Accepts local cards, eZ Cash, FriMi, and bank transfers. Easy to integrate.
- **Stripe** — Best for international customers. Excellent documentation and a free test mode.
- **PayPal** — Worldwide trust. Customers pay without entering card details.

What changes in your code

You will **remove the card form** from **index.php** and replace it with the provider's button. The provider opens its own secure page, takes the card details, and sends the customer back to your **processing.php** with a confirmation token. Your job becomes only to verify that token, then continue exactly as we did in this tutorial.

Important: Once you accept real payments, you must add an SSL certificate to your site so the URL starts with **https://**. Most hosts give one for free with Let's Encrypt.

Final Checklist

	Item
[ ]	All three PHP files are in the same folder
[ ]	downloads/tutorial.pdf exists and opens normally
[ ]	session_start() appears at the top of every PHP file
[ ]	Form action points to processing.php with method POST
[ ]	payment-done.php blocks visitors who did not pay
[ ]	Only the last 4 digits of the card are stored anywhere
[ ]	Site uses HTTPS before going live with real payments

Next tutorial: Add a real payment provider — PayHere step by step.

Find more beginner tutorials at **egotechworld.com**