Perfect—let’s lock in **Option 1 (dev → local PHP)** with a clean, consistent setup. I’ll give you the exact files/contents for:

* Frontend: .env, .env.development, .env.production, vite.config.js, src/api/index.js
* Backend (PHP): .env.example, local .env (dev), server .env (prod), and a config.php loader

⚠️ Security note: your RDS creds were posted in plain text. Rotate that DB password in AWS and update the server .env.

**Frontend (Vite/React) — files at project root**

**1) .env (shared, safe defaults — no secrets)**

VITE\_APP\_NAME=StockLoyal

VITE\_BASE\_PATH=/stockloyal-pwa/

# Default to /api so prod-like paths work unless overridden

VITE\_API\_BASE=/api

**2) .env.development (dev points to your local PHP under XAMPP)**

VITE\_API\_BASE=http://localhost/api

If your local Apache runs on a port, use it (e.g., http://localhost:8080/api).

**3) .env.production (prod builds talk to same origin /api)**

VITE\_API\_BASE=/api

**4) vite.config.js (root)**

import { defineConfig } from "vite";

import react from "@vitejs/plugin-react";

export default defineConfig(() => ({

plugins: [react()],

base: process.env.VITE\_BASE\_PATH || "/stockloyal-pwa/",

define: {

// single source of truth is Vite env

\_\_API\_BASE\_\_: JSON.stringify(process.env.VITE\_API\_BASE ?? "/api"),

},

// No proxy needed for Option 1 (dev → local PHP via full URL)

// If you later prefer to always use "/api" in code, add a proxy here.

}));

**5) src/api/index.js (the only way components should call the API)**

// src/api/index.js

export const API\_BASE = \_\_API\_BASE\_\_;

const buildUrl = (endpoint) => {

if (endpoint.startsWith("http")) return endpoint;

// normalize if someone passes "/api/foo.php"

const cleaned = String(endpoint).replace(/^\/?api\/+/, "");

return `${API\_BASE}/${cleaned}`;

};

export async function apiPost(endpoint, body) {

const resp = await fetch(buildUrl(endpoint), {

method: "POST",

headers: { "Content-Type": "application/json" },

body: JSON.stringify(body),

});

if (!resp.ok) throw new Error(`HTTP ${resp.status} ${resp.statusText}`);

return resp.json();

}

export async function apiGet(endpoint) {

const resp = await fetch(buildUrl(endpoint));

if (!resp.ok) throw new Error(`HTTP ${resp.status} ${resp.statusText}`);

return resp.json();

}

**What this gives you**

* **Dev** (npm run dev): API\_BASE = http://localhost/api
* **Prod build** (npm run build): API\_BASE = /api
* Any stray apiPost('/api/foo.php', …) still works due to normalization.
* Components should **only** use apiGet/apiPost (not raw fetch('/api/...')).

**Backend (PHP under C:\xampp\htdocs\api) — secure env handling**

**1) Do not use VITE\_\* here**

Those are frontend-only. The backend should read **server env** or a PHP .env loaded at runtime.

**2) Commit only an example file**

/api/.env.example (committed):

APP\_ENV=development

DB\_HOST=127.0.0.1

DB\_NAME=stockloyal

DB\_USER=root

DB\_PASS=

DB\_PORT=3306

ENCRYPTION\_KEY=change\_me

ENCRYPTION\_IV=change\_me

**3) Local dev secrets (not in Git)**

/api/.env (on your dev machine only; **do not commit**):

APP\_ENV=development

DB\_HOST=127.0.0.1

DB\_NAME=stockloyal

DB\_USER=root

DB\_PASS=

DB\_PORT=3306

ENCRYPTION\_KEY=your\_dev\_key

ENCRYPTION\_IV=your\_dev\_iv

**4) Server secrets (not in Git)**

/api/.env (on EC2 only; **do not commit**):

APP\_ENV=production

DB\_HOST=stockloyal-db.ctms60ci403w.us-east-2.rds.amazonaws.com

DB\_NAME=stockloyal

DB\_USER=admin

DB\_PASS=<ROTATE\_THIS\_AND\_PUT\_NEW\_PASSWORD\_HERE>

DB\_PORT=3306

ENCRYPTION\_KEY=<secure-value>

ENCRYPTION\_IV=<secure-value>

**5) config.php (load env, build PDO)**

If you use vlucas/phpdotenv:

<?php

// /api/config.php

declare(strict\_types=1);

require\_once \_\_DIR\_\_ . '/vendor/autoload.php';

// Load .env if present (local dev / server fallback)

if (class\_exists(\Dotenv\Dotenv::class)) {

\Dotenv\Dotenv::createImmutable(\_\_DIR\_\_)->safeLoad();

}

// Prefer OS envs (production best practice), falling back to $\_ENV

function envv(string $key, ?string $default = null): ?string {

$v = getenv($key);

if ($v === false) $v = $\_ENV[$key] ?? null;

return $v ?? $default;

}

$env = [

'APP\_ENV' => envv('APP\_ENV', 'production'),

'DB\_HOST' => envv('DB\_HOST', '127.0.0.1'),

'DB\_NAME' => envv('DB\_NAME', ''),

'DB\_USER' => envv('DB\_USER', ''),

'DB\_PASS' => envv('DB\_PASS', ''),

'DB\_PORT' => (int) (envv('DB\_PORT', '3306')),

];

$dsn = sprintf('mysql:host=%s;port=%d;dbname=%s;charset=utf8mb4',

$env['DB\_HOST'], $env['DB\_PORT'], $env['DB\_NAME']);

$options = [

PDO::ATTR\_ERRMODE => PDO::ERRMODE\_EXCEPTION,

PDO::ATTR\_DEFAULT\_FETCH\_MODE => PDO::FETCH\_ASSOC,

PDO::ATTR\_EMULATE\_PREPARES => false,

];

try {

$conn = new PDO($dsn, $env['DB\_USER'], $env['DB\_PASS'], $options);

} catch (Throwable $e) {

http\_response\_code(500);

header('Content-Type: application/json');

echo json\_encode(['ok' => false, 'error' => 'DB connection failed']);

exit;

}

Add a tiny health check:

<?php // /api/health.php

header('Content-Type: application/json');

require\_once \_\_DIR\_\_ . '/config.php';

try { $conn->query('SELECT 1'); echo json\_encode(['ok'=>true,'env'=>$env['APP\_ENV']]); }

catch(Throwable $e){ http\_response\_code(500); echo json\_encode(['ok'=>false,'error'=>$e->getMessage()]); }

**Verify flow (quick tests)**

**Dev (Windows/XAMPP):**

1. Visit http://localhost/api/health.php → expect {"ok":true,"env":"development"}
2. Run npm run dev
3. In browser console:
4. import("../src/api/index.js").then(m => console.log(m.API\_BASE))

Expect http://localhost/api.

1. Use the app → all API calls should hit http://localhost/api/...

**Prod (EC2):**

1. https://app.stockloyal.com/api/health.php → {"ok":true,"env":"production"}
2. Build & deploy PWA → API calls go to /api/... (same origin).

**What to commit vs keep private**

**Commit (frontend root):** .env (no secrets), .env.development, .env.production, vite.config.js, src/api/index.js  
**Commit (backend /api):** .env.example, config.php, health.php  
**Do NOT commit:** /api/.env with real credentials (local or server)

**One-liners you asked about**

* **vite.config.js** should not contain VITE\_API\_BASE=/api. That belongs in .env\*. Keep vite.config.js code as shown above.
* **src/api/index.js** is the helper file (JS code), not an .env. It shouldn’t contain VITE\_API\_BASE=/api. It reads the injected \_\_API\_BASE\_\_ (or you can switch to import.meta.env.VITE\_API\_BASE if you prefer—either is fine as long as it comes from .env\*).

Bottom of Form

**CONFIG CHECKOUT**

OK, everything is working on local dev, now when we migrate to production AWS what should we check first. Last night we played with several config files and htaccess, can we make sure those are aligned correctly?

**ChatGPT said:**

Thought for 14s

Nice—since it’s working locally, here’s a tight production checklist for your AWS move. It’s ordered so you can catch “blockers” first, then harden/tune.

**1) DNS + HTTPS**

* Point your domain (e.g. app.yourdomain.com) to the EC2/ALB.
* Get TLS certs (ACM if using ALB, or Let’s Encrypt on the instance).
* Force HTTPS (redirect HTTP→HTTPS).

**2) Build & deploy the Vite app**

* Build once on CI/your machine: npm run build → produces dist/.
* Upload contents of dist/ to your web root, e.g. /var/www/stockloyal-pwa/ (or Bitnami Apache htdocs/stockloyal-pwa/).
* Keep your **API** in /var/www/stockloyal-pwa/api (PHP files).

**3) Environment variables (critical)**

* In production, the React app must call the PHP API over your public site URL, not relative paths.
* Create .env.production **before** building:
* VITE\_API\_BASE=https://app.yourdomain.com/stockloyal-pwa/api

(If you’re serving at the domain root, drop the stockloyal-pwa segment.)

* Rebuild after changing VITE\_API\_BASE.

**4) PHP config.php (DB + prod flags)**

* Confirm RDS host, user, pass, DB name. Use SSL to RDS.
* Example (PDO):
* <?php
* // api/config.php
* $db\_host = 'stockloyal-db.ctms60ci403w.us-east-2.rds.amazonaws.com';
* $db\_name = 'stockloyal';
* $db\_user = 'admin';
* $db\_pass = '\*\*\*\*\*\*\*';
* $options = [
* PDO::ATTR\_ERRMODE => PDO::ERRMODE\_EXCEPTION,
* PDO::ATTR\_DEFAULT\_FETCH\_MODE => PDO::FETCH\_ASSOC,
* PDO::MYSQL\_ATTR\_INIT\_COMMAND => "SET NAMES utf8mb4",
* PDO::MYSQL\_ATTR\_SSL\_CA => \_\_DIR\_\_ . '/rds-ca-bundle.pem', // put AWS RDS bundle here
* PDO::MYSQL\_ATTR\_SSL\_VERIFY\_SERVER\_CERT => true,
* ];
* $dsn = "mysql:host=$db\_host;dbname=$db\_name;charset=utf8mb4";
* $conn = new PDO($dsn, $db\_user, $db\_pass, $options);
* Place rds-ca-bundle.pem in api/ (or a secure path) and reference it as above.

**5) Security Groups / networking**

* EC2: open 80/443 from the internet.
* RDS: **do not** open 3306 to the internet. Allow **only** the EC2 security group (or specific private IP/CIDR).
* Confirm EC2 can reach RDS:
* mysql -h <rds-endpoint> -u admin -p --ssl-ca=/path/rds-ca-bundle.pem stockloyal

(On Bitnami MariaDB client, use mariadb and --ssl-ca.)

**6) Apache (or Nginx) virtual host**

A clean Apache vhost if you’re serving at /stockloyal-pwa path:

<VirtualHost \*:80>

ServerName app.yourdomain.com

DocumentRoot /var/www/stockloyal-pwa

<Directory /var/www/stockloyal-pwa>

Options FollowSymLinks

AllowOverride All

Require all granted

</Directory>

# Optional: send API to PHP-FPM if you use it; otherwise mod\_php handles it

ErrorLog ${APACHE\_LOG\_DIR}/stockloyal-error.log

CustomLog ${APACHE\_LOG\_DIR}/stockloyal-access.log combined

</VirtualHost>

If using HTTPS directly on the instance, add the :443 vhost with your certs and redirect :80 → :443.

**7) .htaccess (SPA routing + API + security)**

Put this in /var/www/stockloyal-pwa/.htaccess (adjust path if different):

# Deny access to sensitive files

<FilesMatch "^(?:\.env|composer\.(?:json|lock)|rds-ca-bundle\.pem|config\.php)$">

Require all denied

</FilesMatch>

# Compression (if mod\_deflate available)

<IfModule mod\_deflate.c>

AddOutputFilterByType DEFLATE text/html text/plain text/xml text/css text/javascript application/javascript application/json

</IfModule>

# Caching: long cache for static assets, short/no cache for HTML & API

<IfModule mod\_expires.c>

ExpiresActive On

ExpiresByType text/html "access plus 0 seconds"

ExpiresByType application/json "access plus 0 seconds"

ExpiresByType text/css "access plus 1 year"

ExpiresByType application/javascript "access plus 1 year"

ExpiresByType image/svg+xml "access plus 1 year"

ExpiresByType image/png "access plus 1 year"

ExpiresByType image/jpeg "access plus 1 year"

</IfModule>

# Force HTTPS (uncomment if TLS is terminated on the instance)

# <IfModule mod\_rewrite.c>

# RewriteEngine On

# RewriteCond %{HTTPS} !=on

# RewriteRule ^ https://%{HTTP\_HOST}%{REQUEST\_URI} [L,R=301]

# </IfModule>

# SPA routing: send non-file, non-dir requests to index.html

<IfModule mod\_rewrite.c>

RewriteEngine On

# Don’t rewrite API requests, let PHP handle /api/\*.php directly

RewriteRule ^api/ - [L]

# If the request is a real file or directory, serve it

RewriteCond %{REQUEST\_FILENAME} -f [OR]

RewriteCond %{REQUEST\_FILENAME} -d

RewriteRule . - [L]

# Otherwise, serve the SPA

RewriteRule . /index.html [L]

</IfModule>

# CORS for API (adjust domain)

<IfModule mod\_headers.c>

SetEnvIf Origin "https?://(www\.)?yourdomain\.com$" AccessControlAllowOrigin=$0

Header set Access-Control-Allow-Origin "%{AccessControlAllowOrigin}e" env=AccessControlAllowOrigin

Header set Access-Control-Allow-Credentials "true"

Header set Access-Control-Allow-Headers "Content-Type, Authorization"

Header set Access-Control-Allow-Methods "GET, POST, OPTIONS"

</IfModule>

**Notes**

* This preserves /api/\*.php routes and sends everything else to the SPA index.html.
* Blocks access to .env, CA bundle, and config.php.
* If you deploy at the domain root (not under /stockloyal-pwa), ensure DocumentRoot and RewriteRule paths are consistent.

**8) PHP hardening**

* In php.ini (or pool conf):
  + display\_errors = Off
  + log\_errors = On
  + error\_log = /var/log/php\_errors.log
  + expose\_php = Off
  + post\_max\_size / upload\_max\_filesize sane.
* In your PHP API scripts:
  + Always header("Content-Type: application/json");
  + Use json\_encode($data, JSON\_UNESCAPED\_SLASHES);
  + Sanitize inputs; use prepared statements (you are via PDO).

**9) File/dir permissions**

* Web root owned by a deploy user; readable by web server.
* api/config.php, rds-ca-bundle.pem readable by web server, not world-writable.
* Never 777.

**10) Health checks (quick wins)**

* curl -I https://app.yourdomain.com/stockloyal-pwa/ → 200 and index.html.
* curl -s https://app.yourdomain.com/stockloyal-pwa/api/get-merchants.php | jq → JSON loads.
* Browser console: no 404s for get-merchants.php, get-wallet.php, etc.
* From EC2:
* mysql -h <rds-endpoint> -u admin -p --ssl-ca=/path/rds-ca-bundle.pem stockloyal -e "SELECT NOW();"
* PHP error log + Apache error log are quiet on page load.

**11) Common gotchas**

* **Wrong VITE\_API\_BASE** (most common): verify the value baked into the prod bundle (open dist/assets/\*.js and search).
* **Rewrites**: if /api is being sent to the SPA, you’ll get “The server… is not valid JSON”. Ensure the RewriteRule ^api/ - [L] line is present.
* **RDS SSL**: without the CA bundle, PDO may fail or connect insecurely—keep PDO::MYSQL\_ATTR\_SSL\_CA set.
* **CORS**: if you end up with a separate domain for API vs app, loosen the CORS origin to that domain.

**12) Optional: headers & performance**

* Add HSTS after confirming HTTPS is solid:
* Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains; preload"
* Gzip/Brotli: enable server-side (Apache mod\_deflate / mod\_brotli) and keep Vite’s asset hashing.
* Consider a CDN for static dist/ assets.

If you share your current .htaccess, vhost, and config.php paths (where they live on the instance), I’ll tailor the exact snippets to your layout.

Top of Form

Thinking

Bottom of Form

ChatGPT can make mistakes.