
BLACKTECHSTARTUP EDUCATION

Build and Launch a Production Web App

Move beyond a demo: design environments, identity, authorization, database migrations, payments, security tests, observability, rollback and a 30-day operating loop before real users depend on the product.

Move beyond a demo: design environments, identity, authorization, database migrations, payments, security tests, observability, rollback and a 30-day operating loop before real users depend on the product.

A demo proves possibility; production proves responsibility

A prototype can succeed while storing everything in one table, using one administrator account and failing whenever the network behaves badly. Production is the point where a customer can lose money, access somebody else's data, get locked out, receive a duplicate charge or depend on a service that disappears. Write a production-readiness document before launch with owners for identity, data, payments, email, logging, backups, security, support and rollback. If nobody owns a failure mode, the founder owns it at 2 a.m.

Define a small number of service promises you can actually operate: which browsers and devices are supported, what happens during maintenance, how quickly password-reset email should arrive, what data is backed up, how support works and what happens if a third-party provider fails. Do not copy an enterprise SLA you cannot meet. Reliability starts with honest boundaries.

Separate environments, secrets and data before launch

Use separate development, test/staging and production environments. Production credentials should not live in source code or a shared chat. Store secrets in the hosting platform or a dedicated secret manager, use least-privilege service accounts, rotate compromised keys and keep different keys for different environments. Never test destructive database operations against live customer data simply because "it is faster."

Treat database structure changes as deployments. Put migrations in version control, test them on production-like data, back up before risky changes and know how to reverse or forward-fix them. A database change that works on an empty local instance can lock a large production table or break older application versions. Practice the sequence before launch.

Treat identity and authorization as separate problems

Authentication answers “Who are you?” Authorization answers “What are you allowed to do?” A logged-in user should not gain access to another customer’s invoice, project or admin function merely by changing an ID in a URL or API request. Enforce permissions on the server for every sensitive object and operation; do not rely on hiding a button in the browser. OWASP’s 2025 Top 10 again places broken access control at the top, which is why this deserves explicit tests.

Create test accounts for every role and tenant. Attempt horizontal attacks—User A requesting User B’s records—and vertical attacks—a normal user calling administrator actions. Test password reset, email change, MFA if used, session expiration, logout and account recovery. The happy path is only half the product.

Build payment systems for duplicate and delayed events

If you accept payments or subscriptions, assume webhooks can arrive twice, late or out of order. Store provider event IDs and make important handlers idempotent so the same event does not create two orders or grant access twice. Separate the provider’s payment state from your internal entitlement state and record transitions in an audit trail. Never mark an order paid because the browser returned to a “success” page; verify payment on the trusted server side.

Test card declines, abandoned checkout, refund, chargeback, subscription cancellation, renewal failure, tax or invoice behavior if applicable, and a user refreshing the page repeatedly. Reconcile the payment provider against your database on a schedule. Money errors destroy trust faster than cosmetic bugs.

Use security standards as test inventories, not badges

OWASP ASVS 5.0 gives builders a structured set of application-security requirements. Use it to build a launch checklist around input handling, session management, access control, cryptography, files, APIs, logging and configuration. The OWASP Top 10 is an awareness list; ASVS is closer to a verification inventory. For an AI-enabled app, also consider AI-specific abuse paths such as prompt injection, unsafe tool actions, data leakage and untrusted retrieved content.

Patch dependencies, lock dependency versions where appropriate, scan for known vulnerabilities, disable debug modes, restrict cross-origin behavior, use secure cookie settings, rate-limit high-risk endpoints and make error messages useful to operators without exposing secrets to users. Security should be testable behavior, not a “secure” badge in marketing copy.

Observability is how you find the failure your user found first

Before launch, instrument server errors, latency, failed logins, payment failures, background-job failures and critical business events such as signup-to-first-value. Give every request or job a traceable identifier where practical. Logs should answer what happened without recording passwords, full tokens or unnecessary personal data. Create alerts only for conditions somebody will act on; thousands of noisy alerts teach a small team to ignore all of them.

Build a support path that captures account identifier, approximate time, browser/device, screenshot if useful and the action the user attempted. Tie support reports to logs. The first production month is not merely customer service; it is the highest-value reliability research you will get.

Launch with rollback, then run a 30-day operating cycle

Run a launch rehearsal: clean signup, login, core workflow, payment if present, emails, permission-denied cases, mobile browser, backup restore, deployment and rollback. Write the rollback command or hosting procedure down before you need it. Release to a limited cohort first when possible, then watch errors and business metrics before widening access.

For the first 30 days, review weekly: crashes/errors, slow endpoints, top support reasons, signup completion, first-value completion, payment reconciliation, security events, backup status and unresolved defects. Fix recurring causes, not individual symptoms. Guide BTS #04 remains the place to decide what customers should pay for; this guide owns the engineering step of making that product dependable enough to use.

Research behind this guide

Use the primary and authoritative sources below to verify current rules, prices, eligibility and program details before acting. Terms can change.

OWASP — Application Security Verification Standard 5.0



OWASP — Top 10:2025



OWASP — Web Security Testing Guide



CISA — Secure by Design



OWASP — API Security Project



Important: This guide is educational information, not individualized legal, tax, investment, employment, security or compliance advice. For decisions with material legal or financial consequences, use a qualified professional who can review your facts.

SHARE THIS GUIDE ↗

POWERED BY: BLACK TECH STARTUP

blacktechstartup.com