APPS BY VIKAS

Meet Vikas



GenzLudo

Ludo with 3D characters and up to 8 players. A first-of-its-kind no-teaming-up feature keeps play fair. Try it - you'll love the difference.

Download on Google Play



Kachu

Too much scrolling straining your neck? Add your addictive apps and complete daily neck exercises to unlock them.

Download on Google Play

00 / PRE-LAUNCH ENGINEERING GUIDE

103 Things Developers Should Know Before Shipping to Production

103 important production checks explained in easy English - Database, Backend, Security, DevOps, Payments & AI

By Vikas Yadav | vikasbuilds.com

Use this list to review your website or app before launch. Mark a check only after it has been tested in a staging or production-like environment.

Database & Data Management

- [ ] **1. Indexes**

  Add database indexes on columns you search or sort often so queries stay fast as data grows.

- [ ] **2. N+1 Queries**

  Avoid running separate database queries inside loops; fetch related data in one or two queries.

- [ ] **3. Connection Pooling**

  Reuse open database connections instead of creating a brand-new connection for every user request.

- [ ] **4. Migrations**

  Track all database table and column changes in versioned scripts instead of editing live tables manually.

- [ ] **5. Migration Rollbacks**

  Always keep a safe undo plan in case a database update breaks your live app.

- [ ] **6. Backups**

  Schedule automatic daily database backups and store copies in a separate, secure location.

Database & Data Management (continued)

- [ ] **7. Testing a Restore**

  Regularly test restoring your backup onto a test server to verify the backup files actually work.

- [ ] **8. Transactions**

  Group multi-step database writes together so either every step succeeds or nothing gets saved.

- [ ] **9. Race Conditions**

  Prevent bugs when two users edit or buy the exact same item at the same millisecond.

- [ ] **10. Row Locking**

  Temporarily lock a database record while updating sensitive balances or inventory counts.

- [ ] **11. Unique Constraints**

  Enforce uniqueness at the database level (like emails or usernames) so duplicates can never sneak in.

- [ ] **12. Foreign Keys**

  Link related tables properly so you do not end up with broken or orphaned records.

- [ ] **13. Soft Deletes**

  Mark records as deleted with a timestamp instead of permanently erasing data you might need later.

- [ ] **14. Pagination**

  Load data in small pages instead of returning thousands of records in a single API response.

Performance & Infrastructure

- [ ] **15. Timezones**

  Store all dates and times in UTC on the server and convert them to local time only in the UI.

- [ ] **16. Caching**

  Save frequently requested data in fast memory (like Redis) to reduce database load.

Performance & Infrastructure (continued)

- [ ] **17. Cache Invalidation**

  Clear or update cached data immediately when underlying values change so users never see stale info.

- [ ] **18. Load Testing**

  Simulate heavy user traffic before launch to find out where your app slows down or crashes.

- [ ] **19. Cold Starts**

  Account for startup delays when serverless functions or idle containers wake up after inactivity.

- [ ] **20. Timeouts**

  Set strict time limits on external API calls so one slow service does not freeze your whole server.

- [ ] **21. Retries with Backoff**

  Retry failed network calls with increasing wait times instead of hammering a struggling server.

- [ ] **22. Idempotency**

  Ensure repeating the same request (like a double-clicked payment) produces the exact same result once.

- [ ] **23. Queues**

  Put heavy tasks into a queue so users get an instant response while work runs behind the scenes.

- [ ] **24. Background Jobs**

  Move slow actions like sending emails, generating PDFs, or resizing videos off the main request thread.

- [ ] **25. Dead Letter Queues**

  Capture background jobs that keep failing so you can inspect and replay them later without losing data.

- [ ] **26. Cron Jobs**

  Schedule recurring tasks (like daily reports or cleanup scripts) and monitor if they fail to run.

Performance & Infrastructure (continued)

- [ ] **27. Webhooks**

  Listen for real-time event notifications from third-party services like payment or auth providers.

- [ ] **28. Webhook Retries**

  Handle duplicate or delayed incoming webhooks safely, as providers resend them if your server is busy.

- [ ] **29. Debouncing**

  Delay firing rapid repeated actions (like search typing) until the user pauses for a moment.

- [ ] **30. Autoscaling**

  Configure servers to automatically add capacity during traffic spikes and scale down when quiet.

- [ ] **31. CDN**

  Serve images, CSS, and JS files from edge servers close to users around the world for fast loading.

- [ ] **32. Image Optimization**

  Compress images and use modern formats (like WebP) so pages load quickly on mobile networks.

- [ ] **33. Bundle Size**

  Keep frontend JavaScript bundles small by removing unused libraries and lazy-loading heavy pages.

- [ ] **34. Error Tracking**

  Use tools (like Sentry) to capture crashes with full stack traces the moment users hit a bug.

- [ ] **35. Structured Logs**

  Write clean JSON logs with user and request IDs so you can search and debug issues fast.

- [ ] **36. Uptime Monitoring**

  Ping your app every minute from outside servers so you know immediately if your site goes offline.

Reliability & System Management

- [ ] **37. Alerts**

  Send urgent notifications (Slack, SMS, email) when error rates spike or servers run out of memory.

- [ ] **38. Health Checks**

  Expose a lightweight endpoint that verifies your app and database are alive and ready for traffic.

- [ ] **39. Error Boundaries**

  Catch UI component crashes gracefully so one broken widget does not turn the whole screen white.

- [ ] **40. Friendly Error Pages**

  Show clear, helpful 404 and 500 pages instead of raw code stack traces when something goes wrong.

- [ ] **41. Status Page**

  Provide a public page where users can check live system status during outages or maintenance.

- [ ] **42. Rollbacks**

  Be able to revert back to the previous working version of your app in one click after a bad deploy.

- [ ] **43. Feature Flags**

  Wrap new features in toggles so you can release them gradually or turn them off without redeploying.

- [ ] **44. Kill Switches**

  Keep emergency switches to disable expensive or failing features instantly during an incident.

- [ ] **45. Circuit Breakers**

  Temporarily stop calling a broken third-party service so your app fails fast instead of waiting.

- [ ] **46. Fallbacks**

  Serve cached or default data when a live service fails so users can still use core parts of the app.

Reliability & System Management (continued)

- [ ] **47. Postmortems**

  Write a blameless summary after every outage explaining what happened and how to prevent it next time.

Security

- [ ] **48. Input Validation**

  Validate and sanitize all user input on the backend—never trust data coming from the browser or app.

- [ ] **49. SQL Injection**

  Always use parameterized queries or an ORM so attackers cannot run malicious database commands.

- [ ] **50. XSS**

  Escape user-generated text before rendering HTML so attackers cannot inject malicious scripts.

- [ ] **51. CSRF**

  Protect state-changing actions with CSRF tokens and SameSite cookies against forged requests.

- [ ] **52. CORS**

  Restrict which domains can call your API instead of leaving cross-origin access open to everyone.

- [ ] **53. SSRF**

  Block user-supplied URLs from accessing internal cloud metadata or private local network addresses.

- [ ] **54. Password Hashing**

  Hash passwords with strong algorithms like bcrypt or Argon2—never store plain-text passwords.

- [ ] **55. Session Expiry**

  Set reasonable expiration times on login sessions and allow users to log out from all devices.

Security (continued)

- [ ] **56. Token Rotation**

  Refresh authentication tokens regularly and revoke compromised tokens immediately.

- [ ] **57. OAuth Scopes**

  Request only the minimum permissions needed when connecting a user's Google, GitHub, or social account.

- [ ] **58. 2FA**

  Support two-factor authentication (authenticator apps or OTPs) to protect user and admin accounts.

Deployment & Development Workflow

- [ ] **59. Webhook Signatures**

  Verify cryptographic signatures on incoming webhooks so attackers cannot send fake payment events.

- [ ] **60. Upload Limits**

  Restrict maximum file upload sizes and allowed file types to prevent storage abuse and server crashes.

- [ ] **61. Dependency Updates**

  Keep packages and frameworks updated to patch known security vulnerabilities promptly.

- [ ] **62. Least Privilege**

  Give each service, API key, and team member only the exact permissions required for their job.

- [ ] **63. Git Branches**

  Protect your main branch and build new features in separate branches to keep production code clean.

- [ ] **64. Code Review**

  Have another developer review pull requests before merging to catch bugs and logic mistakes early.

Deployment & Development Workflow (continued)

- [ ] **65. Tests**

  Write automated unit and integration tests covering your critical flows like signup, checkout, and auth.

- [ ] **66. CI/CD**

  Automate testing and deployments on every push so shipping updates is fast, repeatable, and safe.

- [ ] **67. Staging**

  Test new releases in a staging environment that mirrors production before real users see them.

- [ ] **68. Dev vs Prod**

  Keep development and production databases, API keys, and storage buckets completely separate.

- [ ] **69. Preview Deploys**

  Generate temporary live URLs for pull requests so teammates can test UI changes before merging.

- [ ] **70. Seed Data**

  Prepare clean sample data scripts so new developers and test environments can spin up quickly.

- [ ] **71. Versioning**

  Use clear version numbers (like v1.2.0) for releases and APIs so breaking changes stay predictable.

- [ ] **72. Changelogs**

  Maintain a readable log of new features, fixes, and updates shipped in each release.

- [ ] **73. Infrastructure as Code**

  Define servers, networks, and cloud resources in config files instead of clicking around dashboards.

Deployment & Development Workflow (continued)

- [ ] **74. DNS**

  Configure domain records properly and lower TTL values before migrations for smooth traffic switches.

- [ ] **75. SSL Renewal**

  Ensure HTTPS certificates renew automatically so your site never greets users with security warnings.

Email, Payments & Billing

- [ ] **76. Email SPF + DKIM**

  Set up SPF, DKIM, and DMARC DNS records so password resets and receipts do not land in spam.

- [ ] **77. Stripe Webhooks**

  Update subscription status using verified payment webhooks, not just frontend redirect callbacks.

- [ ] **78. Failed Payments**

  Handle declined cards gracefully with automatic retries, user alerts, and clear grace periods.

- [ ] **79. Refunds**

  Support full and partial refunds cleanly while automatically revoking paid access when needed.

- [ ] **80. Proration**

  Calculate fair partial charges or credits when users upgrade or downgrade plans mid-month.

- [ ] **81. Sales Tax and VAT**

  Collect and calculate regional taxes (GST, VAT, sales tax) accurately based on customer location.

- [ ] **82. Invoices**

  Generate downloadable, compliant PDF invoices and receipts automatically for every transaction.

Email, Payments & Billing (continued)

- [ ] **83. Usage Limits**

  Enforce plan quotas (API calls, seats, storage) accurately so free users cannot exceed paid limits.

User Data & Privacy

- [ ] **84. Account Deletion**

  Let users delete their accounts easily and clean up their personal data across your systems.

- [ ] **85. Data Export**

  Allow users to download a copy of their personal data to comply with privacy regulations.

- [ ] **86. Privacy Policy**

  Publish clear, honest Privacy Policy and Terms of Service pages before collecting user info.

- [ ] **87. Cookie Consent**

  Ask user permission before loading non-essential tracking or advertising cookies where required.

- [ ] **88. Analytics**

  Track key user funnels and drop-off points without leaking sensitive personal information.

- [ ] **89. Accessibility**

  Support keyboard navigation, readable contrast, and screen-reader labels for all users.

- [ ] **90. SEO Basics**

  Add proper page titles, meta descriptions, Open Graph tags, and sitemaps for public pages.

AI Application Development

- [ ] **91. Prompt Injection**

  Protect your AI from malicious user prompts trying to override system rules or leak secret data.

AI Application Development (continued)

- [ ] **92. Token Costs**

  Monitor input and output token usage per user so AI features stay profitable at scale.

- [ ] **93. Spend Caps**

  Set hard monthly budget limits on AI provider dashboards so a bug cannot drain your bank account.

- [ ] **94. Context Limits**

  Trim or summarize long chat histories and documents so requests stay inside model token limits.

- [ ] **95. Model Fallbacks**

  Automatically switch to a backup AI model if your primary provider slows down or goes offline.

- [ ] **96. Streaming**

  Stream AI responses word-by-word so users see instant progress instead of staring at a spinner.

- [ ] **97. Provider Rate Limits**

  Queue requests and handle 429 rate-limit errors smoothly during sudden bursts of AI traffic.

- [ ] **98. Evals**

  Test prompt and model updates against a fixed set of real examples to ensure quality does not drop.

- [ ] **99. Hallucination Checks**

  Ground answers in verified facts or citations and validate structured outputs before showing them.

- [ ] **100. Embeddings**

  Convert text into numerical vectors to power smart semantic search and content recommendations.

- [ ] **101. Vector Search**

  Index and filter embeddings accurately so RAG pipelines retrieve only relevant, authorized context.

AI Application Development (continued)

- [ ] **102. Caching AI Answers**

  Cache responses for repeated or similar prompts to cut latency and save API costs.

Bonus Check

- [ ] **103. Disaster Recovery Drills**

  Backups alone are not enough. You should actually practice recovering your app, secrets, DNS, and data when something goes wrong so your team knows the exact recovery steps under pressure.

Find more free guides at vikasbuilds.com/resources