The AI Code Production Checklist
Dotnet Developers
Draft for review

Security

1. Search the repo, mobile bundle, and prompt history for API keys, passwords, and connection strings, then rotate anything you find.
2. Require a real login on every route that reads or changes a user's data. Do not trust a hidden button.
3. Check authorization on the server for each record. A user must not open another account by changing an id in the URL.
4. Set session cookies to HttpOnly, Secure, and SameSite, and expire them when the user signs out.
5. Confirm password reset and email change cannot hand the account to someone who only knows the address.
6. Limit upload type and size, and store files outside the public web folder.
7. Send SQL and NoSQL queries with parameters. Do not build queries by gluing user text into a string.
8. Encode user text before it is rendered as HTML so a comment cannot run script.
9. Turn off debug pages, stack traces, and default admin logins in the production environment.
10. Set CORS to the real site origins only. Do not leave it on *.
11. Run a dependency vulnerability scan and patch or remove packages with a known critical issue.
12. Scope third-party API keys to the minimum permissions and write down how to rotate them.
13. Verify webhook signatures before you trust the payload.
14. Rate-limit login, signup, password reset, and any public API.
15. Make sure production error pages do not print stack traces, SQL, or environment variables.

Performance

16. Measure the slowest database queries against a realistic amount of data, not an empty local database.
17. Add indexes for the columns you filter, sort, and join on.
18. Paginate every list. Do not return the whole table to the browser.
19. Remove N+1 loops that query the database once per row inside a loop.
20. Cache repeated reads that do not change every request, and set an expiry.
21. Serve images at the size the screen actually uses, in a compressed format.
22. Defer or split JavaScript the first screen does not need.
23. Move email, exports, and AI calls out of the web request and into a background job.
24. Turn on database connection pooling and set a sane pool size.
25. Record a load-time budget for the main screens and check it before launch.

Scalability

26. Load-test at the number of concurrent users you expect in the next six months.

27. Confirm the app can run as more than one instance without users losing their session.
28. Do not keep session or upload state only in the memory of one server.
29. Store uploads in object storage, not on one machine’s disk.
30. Put slow work on a queue so a traffic spike does not freeze the site.
31. Write down host limits: CPU, memory, database connections, and cold starts.
32. Check that the database, not just the web app, stays responsive at the target concurrency.
33. Document how to roll back if the new version fails under traffic.

Reliability and monitoring

34. Add an uptime check on the public URL and send it somewhere a person will see.
35. Send application errors to a log with the request id, and review it after deploy.
36. Take backups of the database and files, and restore one backup onto a scratch environment.
37. Keep development and production data separate. Do not test against the live database.
38. Deploy production through CI. Do not copy files up from a laptop.
39. Expose a health check the host can call without logging in.
40. Store secrets in the host or a vault. Do not print them in the build log.
41. Name who is contacted when production is down, and how.

Code quality and maintainability

42. Make sure a new engineer can explain the folder structure in about ten minutes.
43. Keep each business rule in one place. Do not copy the same check across the UI, API, and database trigger.
44. Handle errors on purpose. Do not catch them and continue as if the write succeeded.
45. Add tests for payments, permissions, and any path that deletes or emails data.
46. Rewrite or delete AI-generated code you cannot explain.
47. Remove unused packages, sample routes, and commented-out blocks before launch.
48. Validate input at the API boundary, even if the form already checks it.
49. Write a short README that says how to run the app locally and how it is deployed.
50. Have a senior engineer read the diff before it goes to production.