LETSBUILD MYAPP

FREE MIGRATION CHECKLIST

The No-Code to Production Code Migration Checklist

The 46 things to verify before, during, and after moving your app off Bubble, Webflow, Glide, Adalo, FlutterFlow, Softr, Airtable, or Retool. Straight from the team that has shipped 200+ products and runs migrations every month.

Work through it yourself, or hand it to whoever runs your migration and make them check every box. Either way, nothing on this list is optional. Every item is here because skipping it has burned somebody.

LETSBUILD MYAPP

1. Before you decide to migrate

Half the apps we assess should stay put for now. Make the call with numbers, not frustration.

- [ ] Pull your last 6 months of platform costs into one number: plan fees, usage overages (workload units, rows, actions), paid plugins, and per-seat charges.
- [ ] Measure real page-load times on your 5 most-used screens with production data, on a phone, not your laptop.
- [ ] List every feature you have postponed or faked because the platform could not do it properly.
- [ ] List every external tool taped on to fill gaps (Zapier, Make, Memberstack, Wized, sheets) and what each costs monthly.
- [ ] Write down what breaks your business if the platform has a bad outage week, and whether that has already happened.
- [ ] Check upcoming diligence events: fundraising, enterprise deals, or acquisition talks that will ask "who owns the code?"
- [ ] Decide the verdict honestly: stay, migrate fully, or migrate only the flows that hurt (hybrid). All three are valid.

2. Inventory what you actually have

Migrations fail on the features everyone forgot existed. Inventory first, estimate second.

- [ ] Export a complete list of pages/screens, including admin views and the ones only two people use.
- [ ] Document every workflow and automation: trigger, steps, and what depends on it downstream.
- [ ] Map the full data model: every table, field, relationship, and which fields are actually populated.
- [ ] List every integration with its direction (in, out, both) and auth method: payments, email, SMS, calendars, CRMs, webhooks.
- [ ] List user roles and what each can see and do. Note anywhere permissions are enforced only by hiding buttons.
- [ ] Count real monthly active users and peak concurrent usage. This sizes your hosting, not vanity signups.
- [ ] Screenshot every screen. It is the cheapest complete spec you will ever produce.

LETSBUILD MYAPP

3. Plan the target platform

The goal is a mainstream stack any developer can work on, not a clever one.

- [ ] Choose boring, hireable technology (for us: React + TypeScript with Supabase or Firebase) and write one sentence on why.
- [ ] Design the new database schema before any UI is built, including the messy fields your no-code tool let you get away with.
- [ ] Decide how users and passwords migrate: auth provider, reset flow, and what happens to sessions on cutover day.
- [ ] Confirm every current integration has a path in the new stack, especially payments and anything with stored credentials.
- [ ] Set the repo up in YOUR GitHub account from day one. You should own the code before the first commit, not after the last invoice.
- [ ] Get hosting accounts (and their billing) created under your email, not your developer's.
- [ ] Agree the definition of feature parity in writing: which screens, which workflows, which reports.

4. Migrate the data safely

Your data is the only part of the app you cannot rebuild. Treat it accordingly.

- [ ] Take a full export of every table before anything else, and store it somewhere the migration cannot touch.
- [ ] Write an explicit field-by-field mapping from old schema to new, including type conversions and default values.
- [ ] Decide the rule for dirty data: duplicate users, orphaned records, half-filled rows. Clean or carry, but decide.
- [ ] Run a full practice import into the new database and count records table by table against the source.
- [ ] Verify relationships survived: pick 20 real records and trace them across every joined table by hand.
- [ ] Plan the delta: how data created between practice import and cutover day gets across.
- [ ] Keep files and images accounted for. Avatars, uploads, and generated PDFs live outside the database and get forgotten.

LETSBUILD MYAPP

5. Run in parallel, then cut over

Zero-downtime migrations are not luck. They are a parallel run with exit criteria.

- [ ] Keep the old app fully live while the new one is built. No half-migrated limbo states.
- [ ] Test the new app against the parity list with real (copied) data, not seed data.
- [ ] Have 3-5 real users run their actual weekly tasks in the new app and log everything that surprises them.
- [ ] Test every payment path with real cards in test mode: new subscription, upgrade, cancel, refund, failed card.
- [ ] Verify every outbound email and notification still sends, and from the right domain.
- [ ] Write cutover-day steps in order, with an owner for each, including the go/no-go check and the rollback plan.
- [ ] Redirect every old URL that ever ranked or was shared: page-level 301s, not a blanket redirect to the homepage.
- [ ] Keep the old platform account alive (read-only if possible) for at least 30 days after cutover.

6. After cutover: own it properly

The migration is done when you could hire any developer tomorrow and they could ship by Friday.

- [ ] Confirm you hold admin on everything: GitHub, hosting, database, DNS, email sending, error monitoring.
- [ ] Turn on error monitoring and uptime alerts that page a human.
- [ ] Confirm automated database backups run and actually restore. Test one.
- [ ] Get a README that lets a new developer run the app locally in under an hour.
- [ ] Re-submit your sitemap in Search Console and watch rankings for 2-3 weeks for redirect misses.
- [ ] Compare month-one infrastructure bills to your old platform costs, and put the difference in writing. That number is why you did this.
- [ ] Cancel the old platform subscription, but only after the 30-day safety window.