STRETCHLEARN · WORKBOOK

No-Code App Building — Workbook

This workbook turns the course into a shipped app. Each section maps to one course module and mixes guided exercises, fill-in worksheets, and checklists so you finish with one published, working app instead of a pile of notes. Keep it open beside Glide, Softr, or Bubble and complete each item as you build rather than all at once.

How No-Code Apps Actually Work

Pick the right platform and scope your idea down to a single core job before you build anything.

Exercise: Label Three Apps as Data, Components, and Actions

Open three apps you already use and dissect them. For each, name the data it stores, point at three components on a screen, and describe one action that happens on tap. This trains your eye to see the four layers every no-code app shares.

• What type of records does this app appear to store in its data layer?

• Which three on-screen components can you name (list, button, image, form)?

• What action fires when you tap the main button, and what does it change?

• Which of the four layers (data, screens, logic, users) is this app's main strength?

Worksheet: Platform Decision Rubric

Answer the rubric for your app idea, then commit to one platform for the whole course. Choose the simplest tool that can do the job, not the most powerful one you might grow into.

App idea in one line

Mobile, tap-first app from a spreadsheet (yes / no)

Web portal or membership site with data in Airtable (yes / no)

Needs custom multi-step logic or a true marketplace (yes / no)

Priority is shipping in days not weeks (yes / no)

Chosen platform (Glide / Softr / Bubble)

One-sentence reason for the choice

Worksheet: Core Job and MoSCoW Scope Cut

Write the one-sentence core job, then sort every feature into Must, Should, Could, or Will-not-have-for-now. Build only the Must list for V1. Keep that list to three to five features.

Core job sentence (this app lets [user] do [thing] so they [outcome])

Must have (V1 only)

Should have (park for V2)

Could have (park indefinitely)

Will not have, for now (explicit exclusions)

Number of must-have features (aim for 3 to 5)

Checklist: Scope Readiness

- [ ] Named exactly one user and one core job for V1
- [ ] Wrote the core job as a single this-app-lets sentence
- [ ] Cut the must-have list to five features or fewer
- [ ] Wrote down explicit will-not-have exclusions so cut features stop creeping back
- [ ] Committed to one platform using the decision rubric

Data Is the Foundation of Your App

Model your tables, fields, and relationships, then clean the data so the app behaves predictably from screen one.

Worksheet: Data Model Planner

Design your data layer before any screen. Create one table per noun in your app, list its fields with the correct type, and note any relationships to other tables. This planner is the single most valuable page in the workbook.

Table 1 name (one noun)

Table 1 fields and their types (text, number, date, single-select, image, checkbox)

Table 2 name (one noun)

Table 2 fields and their types

Join or link table needed for many-to-many (yes / no, name it)

Relationship lines (which table links to which, one-to-many or many-to-many)

Which rollups or lookups you need (e.g. count of bookings per class)

Exercise: Map Your App's Relationships

Take the gym-style example structure and apply it to your own app. Draw a line for every place one record must know about another, then label the shape and decide what to count or look up across it.

• Where does one record link to many others (one-to-many), and on which side does the link live?

• Do you have any many-to-many links that need a join table?

• What value do you need to roll up or count across a relationship?

• Which linked detail do you need to show with a lookup instead of retyping?

Checklist: Pre-Build Data Cleanup Pass

[ ] Every column has one consistent field type with no stray mismatched values
[ ] Categories use single-select options instead of free-typed text
[ ] Duplicate rows and blank junk rows are removed
[ ] Required fields are filled, or a plan exists for blanks
[ ] Every row has a stable unique ID (record ID or added Row ID)
[ ] Prices are numbers and dates are real dates, not text

Designing Screens and Building the App

Turn clean data into list-and-detail screens, add forms that write data back, then arrange and brand it to look finished.

Exercise: Build a Filtered List That Opens a Detail

Build the core list-to-detail pattern in your chosen platform. Bind a list to your main table, set the tap to open a detail screen, and add a filter and sort so only relevant rows show in a sensible order.

• Which table is the list bound to, and what title, subtitle, and image show on each card?

• What filter did you apply (for example, status is Open)?

• What field did you sort by, and in which direction?

• What does the detail screen show when a user taps a card?

Exercise: Add a Form That Writes Data Back

Add a form or Add action that creates a record in your main action table. Auto-fill the logged-in user and the current date, require the essential fields, and confirm a test submission appears as a new row.

• Which table does the form write to, and which inputs does the user fill?

• Which values are auto-filled or hidden (logged-in user, timestamp)?

• Which fields did you mark required so the record cannot be created without them?

• Where does the user land after submitting, and did the new row appear in the data?

Worksheet: Navigation and Screen Map

Plan how your screens connect before you fiddle with layout. List the top-level navigation destinations and, for each screen, name its single primary action. Keep the top level to three to five destinations.

Top-level destinations (3 to 5 max)

Screen 1 name and its one primary action

Screen 2 name and its one primary action

Screen 3 name and its one primary action

Brand colour (hex) and logo file

App icon chosen (yes / no)

Checklist: Looking-Professional Polish

[ ] Main navigation limited to three to five clear destinations
[ ] Every screen has one obvious, visually prominent primary action
[ ] Brand colour, logo, and app icon are applied
[ ] Titles, icons, and spacing are consistent across screens
[ ] Placeholder text and stock images replaced with real content
[ ] Walked through the app on an actual phone as a first-time user

Users, Payments, and Going Live

Add logins and access rules, take payments with Stripe in test mode, then run a launch pass and get real users.

Exercise: Set Up Sign-In and Two Roles

Add email sign-in, then set row owners on a user-specific table and create at least two roles such as Member and Admin. Test by signing in as two different users and confirming each sees only the correct, limited view.

• Which table uses row owners so users see only their own records?

• What roles did you create, and which field stores the role?

• Which screens or components are hidden from the Member role but shown to Admin?

• When you signed in as each user, did the right content appear and the wrong content stay hidden?

Worksheet: Stripe Payment Plan

Plan how your app takes money before wiring it. Decide the charge type, the amount source, and what a successful payment should change in your data. Always run the first purchase in Stripe test mode.

What is being sold (product, booking, or membership)

Charge type (one-time or subscription)

Amount source (fixed price or read from the record)

What a successful payment updates (e.g. set status to Paid, grant access)

Stripe test card purchase completed (yes / no)

Record updated correctly on a paid test purchase (yes / no)

Checklist: Pre-Launch Test Pass

[ ] Signed in as a brand-new user and completed the core job end to end
[ ] Tested every form with blank and unexpected input to confirm validation holds
[ ] Ran a Stripe test purchase and confirmed the record updated correctly
[ ] Checked the app on both a phone and a computer
[ ] Confirmed access rules hide protected content from users who should not see it
[ ] Replaced any remaining placeholder content with the real thing

Exercise: Publish and Recruit Five Real Users

Publish your app to a live link, then share it with five to ten people in your target audience. Watch how they use it and write down every place they hesitate or get stuck. Those stumbles are your V2 roadmap.

• What is the live URL (and custom domain, if set)?

• Who are the five to ten people you shared it with, and why are they the right audience?

• Where did users hesitate, get confused, or get stuck?

• What are the top three fixes those observations point to for V2?

Your Action Plan

1. Dissect three apps into data, components, and actions to train your eye
2. Run the platform decision rubric and commit to Glide, Softr, or Bubble
3. Write your one-sentence core job and cut scope to three to five must-have features
4. Fill the Data Model Planner: one table per noun, correct field types, relationships drawn
5. Run the pre-build cleanup pass so every field is consistent and every row has a stable ID
6. Build a filtered list that opens a detail screen for your main table
7. Add a form that writes a record back, with auto-filled user and required fields
8. Set up navigation, one primary action per screen, and your brand colour, logo, and icon
9. Add email sign-in, row owners, and at least two roles, then test as two users
10. Connect Stripe in test mode, run a full test purchase, and confirm the record updates
11. Run the pre-launch test pass as a brand-new user on both phone and computer
12. Publish to a live link and share with five to ten real users, logging where they get stuck