AI Automation / No-Code Automation

Design database schemas, workflows, and conditional logic for Bubble applications — speeds up development by 3x.
Difficulty: Advanced
Model: GPT-4 / Claude / Gemini
Use Case: Web App Development, Workflow Design
Updated: May 2026
Why This Prompt Exists
Bubble is powerful but has a steep learning curve. Most new builders get stuck on data types, workflows that don’t fire, and conditional logic that’s backward.

You get:

- workflows that run at the wrong time (or don’t run at all)
- data types set up incorrectly (can’t query later)
- privacy rules that block necessary data (or expose too much)
- repeating group filters that don’t work (wrong data source)
- custom states vs. URL parameters confusion

But Bubble apps follow patterns:

- data types: what entities and fields?
- workflows: when does what happen?
- conditions: show/hide based on what?
- privacy: who can see what?
- API: what external data comes in or out?

Without design, Bubble development is trial and error.

This prompt designs structured Bubble application blueprints.

The Prompt

```
Assume the role of a Bubble.io solution architect who designs
application blueprints.

Your task is to design a Bubble app structure for a given use case.

Generate:

1. DATA TYPES (Bubble's database)

| Data Type | Fields | Field Type | Privacy Rule |
|-----------|--------|------------|--------------|
| [type] | [field: type] | text/number/date/file/relation | [who can
see] |

2. WORKFLOW DIAGRAM

   User Action (click button)
   → Step 1: Validate data
   → Step 2: Create/modify thing
   → Step 3: Display result
   → Step 4: Navigate to page

3. WORKFLOW DETAILS (per workflow)

**Workflow: [name]**
```

- Trigger: [button click / page load / schedule / API]
- Conditions: [only run if...]
- Actions:
  1. [Action] → [target] → [parameters]
  2. [Action] → [target] → [parameters]

4. CONDITIONAL LOGIC
   - Element: [which element]
   - Condition: [if X then Y]
   - What changes: [visible / enabled / different data source]

5. PRIVACY RULES
   - For [Data Type]:
     * Rule 1: [condition] → [create/read/update/delete allowed]
     * Rule 2: [condition] → [create/read/update/delete allowed]

6. CUSTOM STATES
   - State name: [name]
   - Type: [text / number / yes-no / list / thing]
   - Used for: [purpose]

7. API INTEGRATION (if any)
   - External API: [name]
   - Workflow step: [call API, handle response, store data]

INPUTS:

App use case:
[E.G., "Freelance project management tool — clients, projects, tasks,

```
invoices"]

Key user actions:
[E.G., "Client signs up, creates project, adds task, marks task
complete, pays invoice"]

Data relationships (entities):
[E.G., "Client → Projects → Tasks" and "Project → Invoice"]

External integrations:
[E.G., "Stripe for payments, Slack for notifications"]

RULES:
- One data type per entity (don't put everything in one type)
- Use privacy rules for row-level security (not just UI hiding)
- Prefer page parameters over custom states for navigation data
- Use repeating group filters over multiple data sources when possible
- Add loading states for workflows that take >1 second
- Test workflows with Schedule API Workflow on list of things
```

How To Use It

- One data type per entity — don’t put everything in a “User” or “Thing” type.
- Use privacy rules for data security, not just UI conditions (hiding a button isn’t security).
- Prefer page parameters over custom states for data passed between pages.
- Add loading states for workflows that take more than 1 second.
- Test workflows on lists using “Schedule API Workflow on a list of things.”

Example Input

**App use case:**
“Client portal where clients can view project status, upload files, and message the team”

**Key user actions:**
“Client logs in, views project dashboard, uploads a file, sends a message, views task status”

**Data relationships:**
“Client → Projects → Tasks, Client → Messages, Project → Files”

**External integrations:**
“Slack for team notifications, Google Drive for file storage”

Why It Works
Most Bubble tutorials teach isolated concepts — data types, then workflows, then conditions — but not how they work together.

This framework improves outcomes by forcing:

- data type specification (what data exists and who can see it)
- workflow mapping (what triggers what actions)
- conditional logic (when UI elements change)
- privacy rule design (data security at the database level)
- state management (custom states vs. page parameters)

**Failure modes this prevents:**

- Privacy rules blocking data that should be visible (too restrictive)
- Privacy rules exposing data to wrong users (not restrictive enough)
- Workflow conditions backward (e.g., “only when empty” vs. “when not empty”)
- Repeating group data source wrong (search for things vs. display list from parent)
- Custom state reset on page navigation (use page parameters instead)

**This improves on:** Scattered Bubble tutorials. Provides integrated blueprint covering data, logic, and security together.

**Related to:** NCA-02 (Airtable Schema) — Bubble's data types are similar; NCA-01 (Stack Selector) — ensures Bubble is the right tool.

## Build Better AI Systems

Subscribe for advanced prompt engineering, AI coding tools, debugging frameworks, and practical strategies for developers and engineers.

Carefully engineered prompts for people doing real work.

### Share this:

- Share on Facebook (Opens in new window) Facebook
- Share on X (Opens in new window) X

See also No-Code Troubleshooter