

```
# ARISE - Workflow (Step-by-Step Build Process)

**This document is your BUILD BLUEPRINT.** Follow it in order. Each step has time estimates, file references, and Claude prompts to use with Perplexity Labs.
```

```
---
```

System Prerequisites & Dependencies

Ensure you have:

- **Node.js v18+** (check: `node --version`)
- **npm** (comes with Node.js; check: `npm --version`)
- **Git** (check: `git --version`)
- **GitHub account** (for repo + Vercel deployment)
- **Supabase account** (free tier)
- **Vercel account** (free tier, linked to GitHub)
- **VS Code** (or preferred editor)
- **Supabase project created** (with email/password auth enabled)

```
---
```

Installation & Initial Setup Checklist

- [] Create GitHub repo: `arise` (public or private)
- [] Create Supabase project at supabase.com
- [] Enable email/password authentication in Supabase Auth settings
- [] Create Storage bucket: `vault_uploads` (private)
- [] Create Vercel account and link to GitHub
- [] Clone repo locally: `git clone https://github.com/[your-username]/arise.git`
- [] Navigate to folder: `cd arise`

```
---
```

Project Setup Phase (Step 1-2: Foundation)

```
### Step 1: Create Vite + React + TypeScript Project
**Time estimate:** 10 minutes
**Files created:** `package.json`, `src/`, build config
```

```
**Command:**  
```bash  
npm create vite@latest arise -- --template react-ts
cd arise
npm install
```
```

```
**Verify:**  
```bash  
npm run dev
Should start dev server at http://localhost:5173
```
```

```
**Git commit:**  
```bash  
git add .
git commit -m "init: create vite react-ts project"
git push origin main
```  
  
---  
  
### Step 2: Add Tailwind CSS & React Router  
  
**Tailwind CSS installation:**  
```bash  
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p
```  
  
**Configure `tailwind.config.js`:**  
```js  
export default {
 content: ["./index.html", "./src/**/*.{js,ts,jsx,tsx}"],
 theme: { extend: {} },
 plugins: [],
};
```  
  
**Update `src/index.css`:**  
```css  
@tailwind base;
@tailwind components;
@tailwind utilities;
```  
  
**React Router installation:**  
```bash  
npm install react-router-dom
npm install zustand
```  
  
**Verify:** No build errors.  
  
**Git commit:**  
```bash  
git add .
git commit -m "feat: add tailwind & react-router"
git push origin main
```  
  
---  
  
### Step 3: Supabase Client Setup & Environment Variables  
  
**Install Supabase client:**  
```bash
```

```

npm install @supabase/supabase-js
```

**Create ` `.env.local` (DO NOT commit) :**
```
VITE_SUPABASE_URL=https://[project-id].supabase.co
VITE_SUPABASE_ANON_KEY=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Get these from Supabase dashboard → Settings → API → Copy the URL and
anon key.

**Create `src/lib/supabaseClient.ts` :**
```typescript
import { createClient } from '@supabase/supabase-js';

const supabaseUrl = import.meta.env.VITE_SUPABASE_URL;
const supabaseAnonKey = import.meta.env.VITE_SUPABASE_ANON_KEY;

export const supabase = createClient(supabaseUrl, supabaseAnonKey);
```

**Git commit:**
```bash
git add src/lib/supabaseClient.ts .env.example
git commit -m "setup: initialize supabase client"
git push origin main
```

**Add to ` `.gitignore` (if not present) :**
```
.env.local
.env.*.local
```

---

## Database Schema Setup Phase (Step 3: Critical)

**This MUST be done before building features.** Create tables + RLS
policies in Supabase.

### Step 3A: Create Database Tables

Go to Supabase dashboard → SQL Editor → Run the following:

```sql
-- 1. profiles table
CREATE TABLE profiles (
 id UUID PRIMARY KEY DEFAULT auth.uid(),
 email TEXT NOT NULL,
 created_at TIMESTAMPTZ DEFAULT now(),
 updated_at TIMESTAMPTZ DEFAULT now()
);
```

```

```

-- 2. modules table (optional; can hardcode in frontend for now)
CREATE TABLE modules (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES auth.users(id) ON DELETE CASCADE,
    module_type TEXT NOT NULL,
    title TEXT NOT NULL,
    order_index INT DEFAULT 0,
    is_collapsed BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMPTZ DEFAULT now(),
    updated_at TIMESTAMPTZ DEFAULT now(),
    UNIQUE(user_id, module_type)
);

-- 3. items table (shared, with module_type discriminator)
CREATE TABLE items (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES auth.users(id) ON DELETE CASCADE,
    module_type TEXT NOT NULL,
    title TEXT NOT NULL,
    notes TEXT,

    -- Projects fields
    status TEXT,
    next_action TEXT,

    -- Money Moves fields
    type TEXT,
    deadline_at TIMESTAMPTZ,
    prep_status TEXT,
    outcome TEXT,

    -- Skills fields
    current_level TEXT,
    last_practiced_at TIMESTAMPTZ,

    -- Cross-module
    last_activity_at TIMESTAMPTZ DEFAULT now(),

    created_at TIMESTAMPTZ DEFAULT now(),
    updated_at TIMESTAMPTZ DEFAULT now()
);

-- 4. timeline_entries table
CREATE TABLE timeline_entries (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES auth.users(id) ON DELETE CASCADE,

    occurred_at TIMESTAMPTZ NOT NULL,
    entry_text TEXT NOT NULL,
    module_type TEXT NOT NULL,
    item_id UUID REFERENCES items(id) ON DELETE SET NULL,
    is_milestone BOOLEAN DEFAULT FALSE,

```

```

    created_at TIMESTAMPTZ DEFAULT now()
);

-- 5. vault_files table
CREATE TABLE vault_files (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES auth.users(id) ON DELETE CASCADE,
    storage_bucket TEXT NOT NULL DEFAULT 'vault_uploads',
    storage_path TEXT NOT NULL,
    file_name TEXT NOT NULL,
    file_type TEXT NOT NULL,
    file_size BIGINT NOT NULL,
    category TEXT,
    related_module_type TEXT,
    related_item_id UUID REFERENCES items(id) ON DELETE SET NULL,
    uploaded_at TIMESTAMPTZ DEFAULT now()
);
```

```

\*\*Verify:\*\* All tables created in Supabase dashboard.

## Step 3B: Enable Row-Level Security (RLS) Policies

Go to Supabase dashboard → Authentication → Policies → Enable RLS for each table:

```

```sql
-- RLS on profiles
ALTER TABLE profiles ENABLE ROW LEVEL SECURITY;
CREATE POLICY user_profiles_access ON profiles
    FOR ALL USING (auth.uid() = id);

-- RLS on modules
ALTER TABLE modules ENABLE ROW LEVEL SECURITY;
CREATE POLICY user_modules_access ON modules
    FOR ALL USING (auth.uid() = user_id);

-- RLS on items
ALTER TABLE items ENABLE ROW LEVEL SECURITY;
CREATE POLICY user_items_access ON items
    FOR ALL USING (auth.uid() = user_id);

-- RLS on timeline_entries
ALTER TABLE timeline_entries ENABLE ROW LEVEL SECURITY;
CREATE POLICY user_timeline_access ON timeline_entries
    FOR ALL USING (auth.uid() = user_id);

-- RLS on vault_files
ALTER TABLE vault_files ENABLE ROW LEVEL SECURITY;
CREATE POLICY user_vault_access ON vault_files

```

```
FOR ALL USING (auth.uid() = user_id);
```

Verify: Each table shows "RLS enabled" in Supabase dashboard.

Core Feature Development Phase (Steps 4-10: Build Order)

Feature 1: Authentication + Route Protection (2 hours)

Goal: Users can sign up / sign in. Protected routes redirect to login.

Files to create:

- `src/pages/Auth.tsx` (sign up / sign in form)
- `src/pages/Dashboard.tsx` (protected; main view)
- `src/hooks/useAuth.ts` (auth state hook)
- `src/components/ProtectedRoute.tsx` (route guard)
- `src/stores/authStore.ts` (Zustand auth state)

Database changes:

- Auth triggers (optional, for onboarding).

API endpoints used:

- Supabase Auth: sign up, sign in, session refresh.

Testing checklist:

- [] Sign up with email + password works.
- [] Sign in works.
- [] Session persists on page refresh.
- [] Logout clears session.
- [] Protected routes redirect unauthenticated users to login.

Time estimate: 2 hours

Claude Prompt for Perplexity Labs:
```
Create a React TypeScript authentication system for ARISE using Supabase email/password auth.

Requirements:


- Sign up page: email, password, confirm password validation
- Sign in page: email, password
- Session management: persist JWT token in browser
- Protected routes: redirect unauthenticated users to login
- Zustand store: manage global user state (user_id, email)
- Components: Auth.tsx (sign up/in form), ProtectedRoute.tsx, useAuth hook



Use Tailwind CSS for styling (calm, minimal design).
Supabase client library for auth operations.
React Router v6 for routing.

```

Output: Complete, working code ready to integrate.
```

```
Git commit:
```bash  
git add src/pages/Auth.tsx src/components/ProtectedRoute.tsx  
src/hooks/useAuth.ts src/stores/authStore.ts  
git commit -m "feat: auth system with supabase email/password"  
git push origin main  
```
```

---

## Feature 2: Dashboard Shell + Module Cards (1.5 hours)

\*\*Goal:\*\* Dashboard renders 5 module cards (Projects, Money Moves, Skills, Vault, ARISE Log). Collapse/expand toggles.

\*\*Files to create:\*\*

- `src/pages/Dashboard.tsx` (main page)
- `src/components/ModuleCard.tsx` (single module card)
- `src/components/ModuleList.tsx` (wrapper)
- `src/stores/uiStore.ts` (collapse/expand state)

\*\*Database changes:\*\* None yet.

\*\*Testing checklist:\*\*

- [ ] Dashboard loads after login.
- [ ] All 5 modules visible.
- [ ] Collapse/expand toggle works.
- [ ] State persists (optional for V1).

\*\*Time estimate:\*\* 1.5 hours

\*\*Claude Prompt:\*\*  
```

Create a Dashboard component for ARISE that displays 5 collapsible module cards.

Modules: Projects, Money Moves, Skills, Vault, ARISE Log

Each module card shows:

- Title + icon
- Summary (e.g., "3 active, 1 paused" for Projects)
- Collapse/expand toggle button

Use Tailwind CSS for card styling (calm, minimal).

React hooks for collapse state (or Zustand).

Responsive layout (mobile: stacked, desktop: flexible).

Output: Complete Dashboard.tsx + ModuleCard.tsx components.
```

\*\*Git commit:\*\*  
```bash

```
git add src/pages/Dashboard.tsx src/components/ModuleCard.tsx  
git commit -m "feat: dashboard with 5 collapsible module cards"  
git push origin main  
```
```

---

### Feature 3: Items CRUD + Module-Specific Forms (3 hours)

\*\*Goal:\*\* Users can create/read/update/delete items. Forms adapt per module\_type.

\*\*Files to create:\*\*

- `src/components/ItemEditor.tsx` (modal form; module-specific fields)
- `src/components/ItemList.tsx` (display items in a module)
- `src/hooks/useItems.ts` (CRUD operations)
- `src/api/itemsAPI.ts` (Supabase queries)

\*\*Database changes:\*\* Insert test data (optional).

\*\*Testing checklist:\*\*

- [ ] Create project: form shows status, next\_action, notes fields.
- [ ] Create money move: form shows type, deadline, prep\_status, outcome fields.
- [ ] Create skill: form shows skill name, current\_level, last\_practiced fields.
- [ ] List items per module.
- [ ] Update item.
- [ ] Delete item.

\*\*Time estimate:\*\* 3 hours

\*\*Claude Prompt:\*\*

```

Create CRUD components for ARISE items. Items are stored in a shared table with module_type discriminator.

Modules: Projects, Money Moves, Skills

Each has different fields:

- Projects: title, status (active/paused/done), next_action, notes
- Money Moves: title, type (internship/hackathon), deadline, prep_status, outcome, notes
- Skills: title, current_level, last_practiced, notes

Components:

- ItemEditor.tsx: modal form that adapts fields based on module_type
- ItemList.tsx: displays items in a module; shows summary + edit/delete buttons
- useItems.ts: hook for CRUD (create, read, update, delete)
- itemsAPI.ts: Supabase PostgREST queries (filtered by user_id + module_type)

Use Tailwind CSS for forms (minimal, calm).

Supabase client for database queries.

Toast notifications for feedback.

Output: Complete, working code.

```
```bash
git add src/components/ItemEditor.tsx src/components/ItemList.tsx
src/hooks/useItems.ts src/api/itemsAPI.ts
git commit -m "feat: items CRUD with module-specific forms"
git push origin main
```
---
```

Feature 4: Log Progress (Timeline Entry + Item Update) (2 hours)

Goal: Hero button "Log Progress" creates a timeline entry + updates item's last_activity_at.

Files to create:

- `src/components/LogProgressModal.tsx` (form: date auto, note, optional link to item)
- `src/api/timelineAPI.ts` (create timeline entry)
- `src/hooks/useLogProgress.ts` (combined create logic)

Database changes: None (schema already includes timeline_entries).

Testing checklist:

- [] "Log Progress" button visible on dashboard.
- [] Modal opens with date auto-filled, note field, item link dropdown.
- [] Clicking "Log" creates timeline entry + updates item.last_activity_at.
- [] UI refreshes; entry appears in timeline + item shows updated activity.

Time estimate: 2 hours

Claude Prompt:

Create LogProgressModal for ARISE.

Behavior:

1. User clicks "Log Progress" button on dashboard.
2. Modal opens with:
 - Date/time (auto-filled to now, but editable)
 - Short note text field (required)
 - Optional dropdown to link to a project/item
3. User types note + optionally selects item.
4. User clicks "Log" button.
5. System:
 - Creates entry in timeline_entries table
 - Updates items.last_activity_at for linked item (if selected)
 - Refreshes dashboard + timeline view

6. Toast shows "Progress logged!"

Components:

- LogProgressModal.tsx: form + logic
- useLogProgress.ts: hook combining timeline insert + item update
- timelineAPI.ts: Supabase queries for timeline

Use Tailwind CSS.

Keep form minimal (date + note + link, that's it).

Output: Complete code.

```

```
Git commit:
```bash  
git add src/components/LogProgressModal.tsx src/api/timelineAPI.ts  
src/hooks/useLogProgress.ts  
git commit -m "feat: log progress (timeline entry + item update)"  
git push origin main  
```
```

---

### Feature 5: Vault Uploads + File Management (2.5 hours)

\*\*Goal:\*\* Users can upload PDF/JPG/PNG/DOCX to Vault. Files are tagged and stored in Supabase Storage.

\*\*Files to create:\*\*

- `src/components/VaultUpload.tsx` (drag-and-drop file picker)
- `src/components/VaultList.tsx` (display files; download/delete)
- `src/api/vaultAPI.ts` (Supabase Storage + metadata CRUD)
- `src/hooks/useVault.ts` (upload/delete logic)

\*\*Database changes:\*\* None (vault\_files table already created).

\*\*Supabase Storage setup:\*\*

- Create bucket: `vault\_uploads` (private).

\*\*Testing checklist:\*\*

- [ ] Upload PDF: file stored, metadata saved.
- [ ] Upload JPG/PNG: works.
- [ ] Upload DOCX: works.
- [ ] File size validation (< 100MB).
- [ ] File type validation (PDF, JPG, PNG, DOCX only).
- [ ] Download file works.
- [ ] Delete file removes storage + metadata.
- [ ] Files tagged by category (optional: link to item).

\*\*Time estimate:\*\* 2.5 hours

\*\*Claude Prompt:\*\*

```

Create Vault file upload/download for ARISE using Supabase Storage + metadata DB.

Components:

- VaultUpload.tsx: drag-and-drop file picker
- VaultList.tsx: list vault files with download/delete actions
- vaultAPI.ts: Supabase Storage upload/download + vault_files metadata CRUD
- useVault.ts: hook for upload/delete

Requirements:

- File types: PDF, JPG, PNG, DOCX only
- Max file size: 100MB
- Files stored in vault_uploads bucket (private access)
- Metadata saved: file_name, file_type, file_size, category, related_item_id
- UI: show file name, size, upload date, download + delete buttons

Use Tailwind CSS (minimal form).

Toast notifications for feedback.

RLS enforces user_id isolation.

Output: Complete, working code.

```

\*\*Git commit:\*\*

```
```bash
git add src/components/VaultUpload.tsx src/components/VaultList.tsx
src/api/vaultAPI.ts src/hooks/useVault.ts
git commit -m "feat: vault file uploads (supabase storage + metadata)"
git push origin main
````
```

---

### Feature 6: Global Timeline View + Filters (2 hours)

\*\*Goal:\*\* Dedicated Timeline view. Displays all timeline entries (newest first). Filter by module\_type.

\*\*Files to create:\*\*

- `src/pages/Timeline.tsx` (timeline page)
- `src/components/TimelineList.tsx` (render entries)
- `src/components/TimelineFilters.tsx` (filter controls)

\*\*Database changes:\*\* None.

\*\*Testing checklist:\*\*

- [ ] Timeline page shows all entries (newest first).
- [ ] Entries show date, text, linked item (if any).
- [ ] Filter by module\_type works (e.g., "Projects only").
- [ ] Clear filters resets.
- [ ] Pagination (if > 50 entries, load more).

\*\*Time estimate:\*\* 2 hours

\*\*Claude Prompt:\*\*  
```  
Create Timeline page for ARISE.

Components:
- Timeline.tsx (page)
- TimelineList.tsx (render entries chronologically; newest first)
- TimelineFilters.tsx (dropdown/chips to filter by module_type)

Behavior:
- Load entries sorted by occurred_at DESC
- Display: date, entry_text, linked item (if item_id present)
- Filter controls: show "All", "Projects", "Money Moves", "Skills", "Vault", "ARISE Log"
- On filter change, refetch filtered list
- Optional: pagination (load 50 at a time)

Use Tailwind CSS.
Supabase queries via timelineAPI.

Output: Complete code.
```

\*\*Git commit:\*\*  
```bash  
git add src/pages/Timeline.tsx src/components/TimelineList.tsx
src/components/TimelineFilters.tsx
git commit -m "feat: global timeline view with filters"
git push origin main
```

---

## Feature 7: Main App Shell + Routing (1 hour)

\*\*Goal:\*\* Wire up all routes. App.tsx orchestrates navigation between Auth, Dashboard, Timeline, Vault, etc.

\*\*Files to create:\*\*  
- `src/App.tsx` (main routing)  
- `src/components/Navbar.tsx` (top navigation)

\*\*Testing checklist:\*\*  
- [ ] Routes work: /login, /dashboard, /timeline, /vault, /settings (optional).  
- [ ] Protected routes redirect to login.  
- [ ] Navigation between pages works.  
- [ ] Navbar shows current user + logout button.

\*\*Time estimate:\*\* 1 hour

\*\*Claude Prompt:\*\*

```

Create App.tsx with React Router routing for ARISE.

Routes:

- / → redirect to /dashboard or /login (based on auth)
- /login → Auth page (sign up / sign in)
- /dashboard → Protected, Dashboard page
- /timeline → Protected, Timeline page
- /vault → Protected, Vault page (or integrate into Dashboard as a section)

Layout:

- Navbar: ARISE logo, current user email, logout button
- Main content area below navbar

Use React Router v6.

Protected routes wrap with ProtectedRoute component.

Output: Complete App.tsx + Navbar.tsx.

```

\*\*Git commit:\*\*

```
```bash
git add src/App.tsx src/components/Navbar.tsx
git commit -m "feat: main app routing + navbar"
git push origin main
```
```

---

## Testing Phase (1-2 hours)

\*\*Manual testing checklist:\*\*

### Authentication

- [ ] Sign up with new email works.
- [ ] Sign in with correct credentials works.
- [ ] Sign in with wrong password fails (error message shown).
- [ ] Logout works.
- [ ] Protected routes redirect to login when unauthenticated.
- [ ] Session persists on page refresh.

### Dashboard & Modules

- [ ] Dashboard loads after login.
- [ ] All 5 modules visible.
- [ ] Collapse/expand each module works.
- [ ] Add item modal opens from each module.

### Items CRUD

- [ ] Create project: form has status, next\_action, notes fields; save works.
- [ ] Create money move: form has type, deadline, prep\_status, outcome; save works.

- [ ] Create skill: form has skill name, current\_level, last\_practiced; save works.
- [ ] Items display in module list.
- [ ] Edit item: open form, change fields, save updates.
- [ ] Delete item: confirm dialog, delete works.

### Log Progress

- [ ] Click "Log Progress" button → modal opens.
- [ ] Date auto-filled; editable.
- [ ] Type short note.
- [ ] Optionally link to item (dropdown).
- [ ] Click "Log" → entry created, item.last\_activity\_at updated.
- [ ] Dashboard/Timeline refresh shows new entry.

### Vault

- [ ] Click Vault module → upload UI appears.
- [ ] Drag PDF file → upload works, file appears in list.
- [ ] Upload JPG, PNG, DOCX (all work).
- [ ] Download file works (file opens/downloads).
- [ ] Delete file → confirm, delete works (file removed from list + storage).
- [ ] File size validation (reject > 100MB).

### Timeline

- [ ] Timeline page shows all entries (newest first).
- [ ] Filter by module\_type works.
- [ ] Entries show date, text, linked item.

### Responsive Design

- [ ] Dashboard layout on mobile (stacked cards).
- [ ] Dashboard layout on desktop (multi-column, optional timeline sidebar).
- [ ] Forms responsive (inputs full-width on mobile, centered on desktop).

### Accessibility

- [ ] Tab through form fields (keyboard navigation).
- [ ] Focus indicators visible on buttons/inputs.
- [ ] Form labels present + associated with inputs.

\*\*Issue tracking:\*\* If bugs found, log them + fix before deployment.

---

## Deployment Phase (30 minutes)

### Step 1: Prepare Environment Variables for Vercel

In Vercel dashboard:

1. Connect GitHub repo.
2. Add environment variables:
  - `VITE\_SUPABASE\_URL` = your Supabase URL
  - `VITE\_SUPABASE\_ANON\_KEY` = your Supabase anon key

```
Step 2: Deploy to Vercel

```bash
git push origin main
# Vercel auto-detects push, builds, deploys
# View deployment at https://arise-[random].vercel.app
```

```

### ### Step 3: Verify Deployment

- [ ] Site loads at vercel.app URL.
- [ ] Sign up / sign in works.
- [ ] Dashboard loads.
- [ ] Create item works.
- [ ] No console errors (check Vercel logs).

### ### Step 4: Custom Domain (Optional)

In Vercel dashboard → Settings → Domains → add custom domain (e.g., arise.com).

---

### ## Post-Launch Checklist

- [ ] Verify live URL works (sign up → use app → features work).
- [ ] Monitor Vercel logs for errors.
- [ ] Monitor Supabase dashboard for quota usage.
- [ ] Share with beta testers (if applicable).
- [ ] Collect feedback.
- [ ] Fix bugs found in production.

---

### ## Build Order Priority (If Scope Cutting Needed)

Must-have (V1 core):

1. Auth
2. Dashboard + Modules
3. Items CRUD
4. Log Progress

Nice-to-have (still V1, but can defer 1-2 weeks):

5. Vault uploads
6. Timeline view

Can remove or defer to V1.5:

- Module reordering
- Timeline auto-milestones
- Vault tagging to items

---

### ## Development Tips

```

Local Development
```bash
npm run dev
# Dev server at http://localhost:5173
# Hot reload on file change
```

Building for Production
```bash
npm run build
# Output in dist/
```

Common Commands
```bash
npm install [package]          # Add dependency
npm run dev                     # Start dev server
npm run build                   # Build for production
npm run preview                 # Preview production build
git add .                       # Stage changes
git commit -m "..."            # Commit with message
git push origin main           # Push to GitHub (triggers Vercel deploy)
```

Debugging
- Supabase dashboard: check tables, RLS policies, logs.
- Vercel dashboard: check build logs, deployment logs.
- Browser console: check for JS errors (F12).
- Supabase client logs: enable via `supabase.auth.onAuthStateChange()` listener.

Maintenance & Future Iteration

After V1 Launch
- Monitor user feedback.
- Fix bugs.
- Optimize performance if slow.
- Add improvements based on usage.

V1.5 Ideas (not V1)
- Module reordering (drag-and-drop).
- Auto-milestones (e.g., project marked "done" → auto-create milestone).
- Vault file tagging.
- Dark mode (UI preference).

V2 Ideas
- Multi-user sharing.
- Monthly reports/summaries.
- Search timeline.
- Export data (CSV/PDF).
- Offline support (service workers).

```

---

## ## Questions During Development?

Reference these docs in order:

1. \*\*System Architecture\*\* – understanding layers, data flow, RLS.
2. \*\*Design Document\*\* – UI/UX decisions, flows, components.
3. \*\*Tech Stack\*\* – stack rationale, API endpoints, env setup.

If issues arise:

1. Check System Architecture "Non-Negotiables" first.
2. Ask: "Does this fix align with architecture?"
3. Fix root cause, not symptoms.
4. Update this Workflow doc if process changes.