README - 8 Week Plan for Cursor

How to Use These Documents with Cursor
Created: January 26, 2025
For: Rocky Web Studio AI-First Development
Status: Week 1 Complete ✅ | Weeks 2-8 Ready

📚 Documents Created for You

1. `8_WEEK_PLAN_CORRECTED.md` (Main Implementation Plan)

Purpose: Complete roadmap from now through 8 weeks
Status: ✅ Corrected and implementation-ready

What it contains:

- Week-by-week breakdown (Weeks 2-8)
- Specific tasks with time estimates
- Files to create/modify
- Acceptance criteria for each task
- Protection rules to prevent breaking changes
- Corrected database schemas (UUID-based, matches existing patterns)
- Implementation notes aligned with current codebase

How to use with Cursor:

- Open the document in Cursor
- Copy/paste a week's tasks into your planning tool
- Use as checklist during implementation
- Reference protection notes before modifying files
- Keep open while coding

Key Corrections Made:

- ✅ Uses existing case studies schema (from `docs/SUPABASE_CASE_STUDIES_SCHEMA.md`)
- ✅ All tables use UUID (consistent with existing tables)
- ✅ Timestamps use TIMESTAMPTZ (not TIMESTAMP)
- ✅ Includes admin authentication notes
- ✅ Matches current codebase patterns

---

2. `CURSOR-PROTECTION-GUIDE.md` (Critical!)

Purpose: Save you from accidentally breaking working features
Status: ✅ Complete protection guide

What it contains:

- List of untouchable files (currently live)
- What NOT to do with each file
- Testing procedures before every commit
- Recovery instructions if something breaks
- Quick command reference
- Emergency recovery procedures

How to use with Cursor:

- **Read this FIRST** before modifying any existing code
- Before touching any file, check if it's in the "DO NOT MODIFY" list

- Use the testing checklist before every git commit
- Reference the recovery section if something breaks
- Keep this window open while coding

---

3. `8_WEEK_PLAN_ANALYSIS.md` (Deep Dive Analysis)

Purpose: Detailed analysis of plan discrepancies and recommendations
Status: ✅ Analysis complete

What it contains:

- Comprehensive analysis of original plan vs current codebase
- All discrepancies found
- Recommendations by priority (High/Medium/Low)
- Questions for clarification
- Codebase alignment notes

How to use:

- Reference if you want to understand why corrections were made
- Review recommendations before making decisions
- Use as technical reference

---

4. `8_WEEK_PLAN_SUMMARY.md` (Executive Summary)

Purpose: Quick reference for key findings
Status: ✅ Summary complete

What it contains:

- Quick findings (what's correct, what needed fixing)
- Key recommendations
- Critical files status
- Questions for clarification
- Next steps

How to use:

- Quick reference before starting work
- Share with team/engineers for alignment
- Use for planning discussions

🚀 Step-by-Step Workflow

Before Starting Each Week

1. **Open** `8_WEEK_PLAN_CORRECTED.md`
    - Read the specific week's tasks
    - Understand what files you'll create vs modify
2. **Check** `CURSOR-PROTECTION-GUIDE.md`
    - Verify any existing files you'll touch
    - Read the protection rules for those files
3. **Create a git branch:**

    ```
    git checkout -b week-X-feature-name
    ```

Example: `git checkout -b week-4-case-studies-db`

While Implementing Each Task

1. **Create new files freely** (new files are safe)
2. **If modifying existing files:**
   - Check if it's in the protection guide
   - Create backup: `git stash` or backup branch
   - Make changes carefully
   - Test: `npm run dev`
   - Verify no errors
3. **Follow the acceptance criteria** in the plan
4. **Run testing checklist** before committing (see below)

Before Each Commit

1. **Run TypeScript check:**

   ```
   npm run build
   ```

   Expected: No errors
2. **Run local development:**

   ```
   npm run dev
   ```

   Expected: Server starts without errors
3. **Test the feature** you modified
   - Visit the page/feature
   - Test functionality
   - Verify it works as expected
4. **Check browser console** (F12 → Console)
   - Expected: No red errors (warnings usually okay)
5. **If modified layout/styles:**
   - Run Lighthouse (F12 → Lighthouse → Generate Report)
   - Check: Accessibility >90, Performance >91
6. **Review git changes:**

   ```
   git status
   git diff
   ```

   Verify: What files are changing? Are any protected files modified?
7. **Then commit:**

   ```
   git commit -m "feat(scope): description"
   ```

Format: `type(scope): description`
Types: `feat` , `fix` , `docs` , `style` , `refactor` , `test` , `a11y`

8. **Then push:**

```
git push origin week-X-feature-name
```

📋 Quick Reference

Week 2 (Next - Testing & Refinement)

Tasks: Testing, performance optimization, documentation
Time: 7 hours total
Risk Level: 🟢 Low (mostly reading/testing/documentation)
Files: Only documentation updates
Start with: `8_WEEK_PLAN_CORRECTED.md` → Week 2.1

Key Tasks:

- Manual testing & validation (3h)
- Performance optimization (2h)
- Documentation review & updates (2h)

Week 3 (Case Study - AI Assistant)

Tasks: Second case study development
Time: 7 hours total
Risk Level: 🟢 Low (creating new files)
Files: New `case-studies/ai-assistant/` files + screenshots
Start with: `8_WEEK_PLAN_CORRECTED.md` → Week 3.1

Key Tasks:

- Planning & documentation (2h)
- Case study development (4h)
- PDF generation (1h)

Weeks 4-8 (Database Features, Admin UI, Optimization)

Tasks: Case studies database, admin UI, testimonials, leads, tender, optimization
Time: 30+ hours total
Risk Level: 🟡 Medium-High (creating many new features)
Files: Mix of new files and some modifications
Start with: `8_WEEK_PLAN_CORRECTED.md` → Week 4+

Key Tasks:

- Week 4: Database schema + admin UI (10h)
- Week 5: Testimonials system (6h)
- Week 6: Contact forms & lead capture (5h)
- Week 7: Tender submission (3h)
- Week 8: Optimization & scaling (6h)

✅ Critical Rules

1. Never Modify These Without Extreme Care

🔴 **Critical Files (DO NOT BREAK):**

- `app/api/ai-assistant/route.ts` - Chat API endpoint
- `components/AIAssistantWidget.tsx` - Chat widget component
- `lib/claude.ts` - Claude API integration
- `app/layout.tsx` - Widget integration point
- `supabase/migrations/*.sql` - Database migrations (never revert)
- `.env.local` / Vercel env vars - Secrets and configuration

If you absolutely must modify these:

1. Read `CURSOR-PROTECTION-GUIDE.md` (the specific file section)
2. Create a backup branch: `git checkout -b backup-feature-name`
3. Make one small change at a time
4. Test immediately: `npm run dev` → test feature
5. Only then continue

---

2. Always Test Before Commit

Mandatory Testing Checklist:

```
# 1. TypeScript check
npm run build

# 2. Local development
npm run dev

# 3. Browser testing
# - Visit modified page/feature
# - Test functionality
# - Check F12 → Console (no errors)

# 4. If modified layout/styles
# - Run Lighthouse (F12 → Lighthouse)
# - Check scores (Accessibility >90, Performance >91)

# 5. If modified anything global
# - Test chat widget (visit homepage, click widget, send message)

# 6. Review changes
git status
git diff

# 7. Then commit
git commit -m "type(scope): description"
```

---

3. If Something Breaks

Quick Recovery Steps:

1. Check `CURSOR-PROTECTION-GUIDE.md` → "Emergency Recovery" section
2. Or: `git diff` (see what changed)
3. Or: `git checkout filename` (revert specific file)
4. Or: `git reset HEAD~1` (undo last commit, keeps changes)

5. Or: Restore from backup branch
6. Contact for help if unsure

Common Issues:

- Chat widget disappeared → Check `app/layout.tsx` (missing `<AIAssistantWidget />`?)
- Chat not responding → Check `app/api/ai-assistant/route.ts` (API broken?)
- TypeScript errors → Check `types/supabase.ts` (types outdated?)
- Lighthouse dropped → Check colors/contrast, images, semantic HTML

🎯 Success Formula

Week 1 (Already Done) = Understanding the Codebase ✅
Weeks 2-8 = Building New Features Safely

Following this plan means:

- ✅ Zero accidental breakage of working features
- ✅ Smooth deployments every time
- ✅ Type-safe code (100% TypeScript)
- ✅ Accessible design (WCAG 2.1 AA)
- ✅ Excellent performance (Lighthouse >91)
- ✅ Complete features every week

📞 Quick Help

"I'm about to modify file X, is it safe?"

→ Check `CURSOR-PROTECTION-GUIDE.md` → Search for filename
→ If marked 🔴 = be very careful, read section first
→ If marked 🟡 = be cautious, test thoroughly
→ If not there = probably safe, but still test

"I got an error during npm run build"

→ Read the error message carefully
→ Check `CURSOR-PROTECTION-GUIDE.md` → "TypeScript Build Errors" section
→ Most likely: removed an import or changed a type
→ Fix: Add missing import or fix type

"The chat widget disappeared!"

→ Check `CURSOR-PROTECTION-GUIDE.md` → "Chat Widget Disappeared" section
→ Most likely: removed `AIAssistantWidget` from `app/layout.tsx`
→ Fix: `git checkout app/layout.tsx`

"Lighthouse score dropped"

→ Check `CURSOR-PROTECTION-GUIDE.md` → "Lighthouse Score Dropped" section
→ Most likely: changed colors or added heavy images
→ Fix: Run Lighthouse again to see which metric dropped, then fix

---

"I modified a protected file and something broke!"

→ Check `CURSOR-PROTECTION-GUIDE.md` → "Emergency Recovery" section
→ Match your error to the symptom

→ Follow the fix instructions
→ Verify: `npm run dev` → test feature

📖 How to Read the Main Plan

Each week section in `8_WEEK_PLAN_CORRECTED.md` is formatted the same way:

```
WEEK X: FEATURE NAME
Dates: [ADJUST BASED ON TIMELINE] | Status: READY TO START

### X.1: Task Name (time estimate)
Goal: What we're trying to accomplish
Files to Create: List of new files
Files to Modify: List of existing files
(If modifying: reference CURSOR-PROTECTION-GUIDE.md)

Detailed instructions...

Output: What's complete when done
```

Key Sections:

- **Goal:** What you're trying to achieve
- **Files to Create:** New files (safe to create)
- **Files to Modify:** Existing files (check protection guide first)
- **Output:** What success looks like

🎓 Learning Tips

For Cursor Users

1. **Paste the specific week's tasks into Cursor**
    - Let Cursor help you implement following the plan
2. **Before accepting Cursor's code changes:**
    - Check if it modifies protected files
    - Verify it follows the acceptance criteria
    - Test before committing
3. **Use Cursor's AI features:**
    - Ask Cursor to explain protected files before modifying
    - Use Cursor to generate new files (safer than modifying existing)
    - Let Cursor help with TypeScript types

For Developers

1. **Use `8_WEEK_PLAN_CORRECTED.md` as your sprint plan**
    - Break down week into daily tasks
    - Track progress with checkboxes
2. **Use `CURSOR-PROTECTION-GUIDE.md` before writing code**
    - Know what's safe to modify
    - Understand protection rules

- Know how to recover if something breaks

3. **Use the testing checklist before every commit**
   - Don't skip steps
   - Verify everything works
   - Then commit with confidence

🔄 Document Relationships

```
8_WEEK_PLAN_CORRECTED.md (Main Plan)
    ↓ (references)
CURSOR-PROTECTION-GUIDE.md (Protection Rules)
    ↓ (protects)
Critical Files (app/api/ai-assistant/route.ts, etc.)

8_WEEK_PLAN_ANALYSIS.md (Deep Analysis)
    ↓ (informs)
8_WEEK_PLAN_CORRECTED.md (Corrected Plan)

8_WEEK_PLAN_SUMMARY.md (Quick Reference)
    ↓ (summarizes)
All Documents
```

Workflow:

1. Start with `8_WEEK_PLAN_SUMMARY.md` (quick overview)
2. Read `8_WEEK_PLAN_CORRECTED.md` (detailed plan)
3. Reference `CURSOR-PROTECTION-GUIDE.md` (before modifying files)
4. Use `8_WEEK_PLAN_ANALYSIS.md` (if you want technical details)

✨ Final Notes

You've completed Week 1 successfully. ✅

These documents ensure the next 7 weeks are equally successful by:

- ✅ Giving clear direction (what to build)
- ✅ Protecting what works (don't break these files)
- ✅ Providing safety nets (recovery instructions)
- ✅ Setting quality standards (testing requirements)

Follow the plan + Use the protection guide = Success. 🚀

📊 Status Dashboard

Week	Status	Time Estimate	Risk Level	Start Document
Week 1	✅ Complete	7-11h	🟢 Low	N/A
Week 2	🔵 Ready	7h	🟢 Low	`8_WEEK_PLAN_CORRECTED.md` → Week 2
Week 3	🔵 Ready	7h	🟢 Low	`8_WEEK_PLAN_CORRECTED.md` → Week 3
Week 4	🔵 Ready	10h	🟡 Medium	`8_WEEK_PLAN_CORRECTED.md` → Week 4
Week 5	🔵 Ready	6h	🟡 Medium	`8_WEEK_PLAN_CORRECTED.md` → Week 5

Week 6	🔵 Ready	5h	🟡 Medium	`8_WEEK_PLAN_CORRECTED.md` → Week 6
Week 7	🔵 Ready	3h	🟢 Low	`8_WEEK_PLAN_CORRECTED.md` → Week 7
Week 8	🔵 Ready	6h	🟡 Medium	`8_WEEK_PLAN_CORRECTED.md` → Week 8

Legend:

- ✅ Complete
- 🔵 Ready to start
- 🔴 Needs attention
- 🟢 Low risk
- 🟡 Medium risk
- 🔴 High risk

Status: Ready to Start Week 2
Confidence: 95% (corrected plan aligns with codebase)
Next Action: Read Week 2 section of `8_WEEK_PLAN_CORRECTED.md`

YOU'VE GOT THIS. 💪