Track Development Plan - Complete Roadmap

Total Timeline: 10 weeks (280-320 hours) **Framework:** Next.js 14 (App Router) **Database:** Supabase (PostgreSQL) **Target:** Launch-ready SaaS product

Category Labels:

- **[UI]** = Frontend work (React components, styling, user interactions)
- **[Backend]** = Server Actions (business logic, data operations)
- **[Supabase]** = Database setup, RLS policies, storage configuration

---

📋 Project Overview

Track is a service delivery workspace for agencies, freelancers, and consultants. It replaces scattered tools (project management, client communication, file sharing) with a unified workspace where planning, creation, collaboration, and client delivery all happen in one place.

Core Value Proposition:

- Replace 50-tab chaos with one workspace
- Real-time collaboration for teams
- Professional client portal for service delivery
- AI assistant for natural language control

🏗️ Technical Architecture

User Interaction
↓
Next.js Frontend (React Components)
↓
Server Actions (Business Logic)
↓
Supabase (Database + Auth + Storage + Real-time)
↓
PostgreSQL Database

Key Patterns:

- Server Actions for all data mutations
- RLS (Row Level Security) for multi-tenancy
- Real-time subscriptions for collaboration
- Optimistic UI updates for responsiveness

PHASE 0: FOUNDATION SETUP

Duration: Week 1 (12-15 hours) **Goal:** Get development environment working, deploy hello-world, establish database foundation

---

TASK 0.1: Project Initialization

Goal: Create Next.js project with proper configuration **Duration:** 2 hours

Subtasks:

0.1.1: Create Next.js Project (30 min) - [UI]

- Run create-next-app with TypeScript, Tailwind, App Router
- Verify development server runs
- Test default page loads

0.1.2: Configure Git (15 min) - [UI]

- Initialize git repository
- Create .gitignore
- Initial commit
- Connect to GitHub

0.1.3: Install Essential Dependencies (15 min) - [UI]

- Supabase client libraries
- Icon library (lucide-react)
- Date handling utilities
- Class name utilities

0.1.4: Setup shadcn/ui (30 min) - [UI]

- Initialize shadcn/ui
- Install core components: button, input, card, dialog, table, dropdown-menu
- Verify component library is accessible

0.1.5: Configure Code Quality Tools (30 min) - [UI]

- Setup Prettier for formatting
- Configure ESLint rules
- Add format scripts
- Test formatting works

Deliverable: Project runs locally, git initialized, core dependencies installed

---

TASK 0.2: Supabase Setup

Goal: Create Supabase project and establish database connection **Duration:** 3 hours

Subtasks:

0.2.1: Create Supabase Project (15 min) - [Supabase]

- Create new Supabase project
- Choose appropriate region
- Save database credentials securely

0.2.2: Get Connection Credentials (5 min) - [Supabase]

- Copy Project URL
- Copy anon public key
- Copy service_role secret key

0.2.3: Configure Environment Variables (10 min) - [Backend]

- Create .env.local file
- Add Supabase URL and keys
- Restart dev server

0.2.4: Create Supabase Client Utilities (45 min) - [Backend]

- Build browser client helper
- Build server client helper
- Handle cookie-based sessions

0.2.5: Test Connection (15 min) - [Backend]

- Create test query
- Verify connection from both client and server
- Confirm no errors

Deliverable: Supabase connected, can query from Next.js

TASK 0.3: Authentication Setup

Goal: Implement email/password authentication with protected routes **Duration:** 4 hours

Subtasks:

0.3.1: Enable Email Auth in Supabase (15 min) - [Supabase]

- Enable email provider in Supabase dashboard
- Disable email confirmation for development
- Configure redirect URLs

0.3.2: Create Auth Middleware (1 hour) - [Backend]

- Build middleware to check authentication
- Redirect unauthenticated users to login
- Allow authenticated users through
- Configure protected route patterns

0.3.3: Build Login Page (1.5 hours) - [UI] + [Backend]

- [UI] Create login page with email/password form (45 min)
- [Backend] Build Server Action to handle login (45 min)
- Handle errors and display messages
- Redirect to dashboard on success
- Add link to signup page

0.3.4: Build Signup Page (1.5 hours) - [UI] + [Backend]

- [UI] Create signup page with email/password/name form (45 min)
- [Backend] Build Server Action to handle registration (45 min)
- Handle errors and display messages
- Redirect to dashboard on success
- Add link to login page

0.3.5: Create Auth Callback Route (30 min) - [Backend]

- Handle Supabase auth callback

- Exchange code for session
- Redirect appropriately

Deliverable: Users can sign up, log in, and are redirected to dashboard

---

TASK 0.4: Initial Deployment

Goal: Deploy to Railway, verify production environment works **Duration:** 2 hours

Subtasks:

0.4.1: Prepare for Deployment (30 min) - [Backend]

- Run production build locally
- Fix any build errors
- Verify all dependencies are in package.json

0.4.2: Create Railway Project (15 min) - [Backend]

- Sign up for Railway
- Create new project from GitHub repo
- Connect repository

0.4.3: Configure Environment Variables (15 min) - [Backend]

- Add all environment variables in Railway
- Use production Supabase credentials

0.4.4: Configure Build Settings (15 min) - [Backend]

- Verify Next.js is detected
- Confirm build and start commands
- Set Node version if needed

0.4.5: Deploy and Test (45 min) - [Backend]

- Trigger deployment
- Monitor build logs
- Access deployment URL
- Test authentication in production

Deliverable: Track is live at public URL, authentication works

---

TASK 0.5: Database Schema Creation

Goal: Create all database tables with proper relationships and constraints **Duration:** 4 hours

Subtasks:

0.5.1: Create Workspaces Table (20 min) - [Supabase]

- Fields: id, name, owner_id, created_at, updated_at
- Primary key on id
- Foreign key to auth.users

0.5.2: Create Workspace_Members Table (20 min) - [Supabase]

- Fields: id, workspace_id, user_id, role, created_at
- Role enum: owner, admin, teammate
- Unique constraint on workspace_id + user_id
- Foreign keys to workspaces and auth.users

0.5.3: Create Clients Table (20 min) - [Supabase]

- Fields: id, workspace_id, name, email, company, phone, address, website, notes, created_at
- Foreign key to workspaces
- Index on workspace_id

0.5.4: Create Projects Table (25 min) - [Supabase]

- Fields: id, workspace_id, client_id, name, status, due_date, created_at, updated_at
- Status enum: not_started, in_progress, complete
- Due_date can be date or text ("quarterly", "weekly")
- Foreign keys to workspaces and clients
- Indexes on workspace_id and client_id

0.5.5: Create Tabs Table (20 min) - [Supabase]

- Fields: id, project_id, parent_tab_id, name, position, created_at
- parent_tab_id nullable (null = top-level)
- Foreign key to projects
- Self-referencing foreign key for parent_tab_id
- Indexes on project_id and parent_tab_id

0.5.6: Create Blocks Table (30 min) - [Supabase]

- Fields: id, tab_id, parent_block_id, type, content, position, created_at, updated_at
- parent_block_id nullable (for section blocks)

- Type enum: text, link, embed, image, video, pdf, timeline, divider, list, table, task, section, stripe_payment
- Content as JSONB
- Foreign key to tabs
- Self-referencing foreign key for parent_block_id
- Index on tab_id

0.5.7: Create Block_Highlights Table (15 min) - [Supabase]

- Fields: id, block_id, color, created_by, created_at
- Color enum: yellow, green, red, blue
- Foreign keys to blocks and auth.users

0.5.8: Create Comments Table (20 min) - [Supabase]

- Fields: id, target_type, target_id, user_id, text, created_at, updated_at, deleted_at
- target_type enum: block, tab, project
- Foreign key to auth.users
- Indexes on target_type and target_id

0.5.9: Create Tab_Shares Table (20 min) - [Supabase]

- Fields: id, tab_id, shared_with_email, permissions, access_token, password_hash, created_by, created_at
- permissions enum: view, comment, edit
- access_token unique
- Foreign keys to tabs and auth.users

0.5.10: Create Files Table (20 min) - [Supabase]

- Fields: id, workspace_id, uploaded_by, file_name, file_size, file_type, storage_path, created_at
- Foreign keys to workspaces and auth.users
- Index on workspace_id

0.5.11: Create File_Attachments Table (15 min) - [Supabase]

- Fields: id, file_id, block_id, display_mode, created_at
- display_mode enum: inline, linked
- Foreign keys to files and blocks
- Index on block_id

0.5.12: Create Payments Table (20 min) - [Supabase]

- Fields: id, workspace_id, project_id, client_id, amount, status, due_date, stripe_payment_link, notes, created_at

- status enum: pending, paid, overdue
- Foreign keys to workspaces, projects, clients
- Indexes on workspace_id and project_id

Deliverable: All database tables created with proper structure

---

TASK 0.6: Row Level Security Setup

Goal: Configure RLS policies to enforce multi-tenancy and permissions **Duration:** 2 hours

Subtasks:

0.6.1: Enable RLS on All Tables (15 min) - [Supabase]

- Enable RLS for all tables created

0.6.2: Workspace-Level Policies (30 min) - [Supabase]

- Users can SELECT workspaces they're members of
- Users can INSERT workspaces
- Owners/admins can UPDATE their workspaces
- Owners can DELETE workspaces

0.6.3: Member-Level Policies (20 min) - [Supabase]

- Users can SELECT members in their workspaces
- Owners/admins can INSERT new members
- Owners/admins can UPDATE member roles
- Owners/admins can DELETE members

0.6.4: Content-Level Policies (40 min) - [Supabase]

- Users can SELECT content in their workspaces
- Teammates can INSERT content
- Teammates can UPDATE content
- Only admins can DELETE content

0.6.5: Client Portal Policies (15 min) - [Supabase]

- Clients can SELECT tabs shared with them via access_token
- Clients can INSERT comments if permission allows
- Clients can UPDATE blocks if permission is "edit"

Deliverable: Multi-tenancy enforced at database level

TASK 0.7: Test Data Setup

Goal: Create sample data for development and testing **Duration:** 30 minutes

Subtasks:

0.7.1: Create Test Workspace (5 min) - [Supabase]

- Insert test workspace
- Add yourself as owner

0.7.2: Create Test Clients (5 min) - [Supabase]

- Insert 2-3 test clients

0.7.3: Create Test Projects (10 min) - [Supabase]

- Insert 3-5 test projects
- Link to test clients
- Vary statuses and due dates

0.7.4: Create Test Tabs and Blocks (10 min) - [Supabase]

- Create tabs for one project
- Add various block types
- Create nested sub-tabs

Deliverable: Sample data exists for UI development

PHASE 1: PROJECTS TABLE & DASHBOARD

Duration: Week 2 (38-43 hours) **Goal:** Build the main dashboard with projects table

---

TASK 1.1: Workspace & Member Logic

Goal: Build functions to manage workspaces and team members **Duration:** 7 hours

Subtasks:

1.1.1: Create Workspace Server Action (1.5 hours) - [Backend]

- Function to create new workspace
- Add creator as owner in workspace_members
- Validate user can only create if they don't have one yet
- Return workspace object

1.1.2: Get User Workspaces Server Action (1 hour) - [Backend]

- Query all workspaces user is member of
- Include role information
- Return array of workspaces with membership data

1.1.3: Invite Member Server Action (2 hours) - [Backend]

- Validate inviter has admin/owner permissions
- Check if invitee email exists
- Create workspace_member record
- Set permissions based on role
- Send invitation email placeholder

1.1.4: Update Member Role Server Action (1 hour) - [Backend]

- Validate requester is owner/admin
- Update member's role
- Prevent demoting last owner

1.1.5: Remove Member Server Action (1 hour) - [Backend]

- Validate requester is owner/admin
- Cannot remove last owner
- Delete member record

1.1.6: Test All Workspace Functions (30 min) - [Backend]

- Test each function with various scenarios
- Verify permissions are enforced

Deliverable: Complete workspace and member management logic

TASK 1.2: Client Management Logic

Goal: Build functions to manage clients **Duration:** 5 hours

Subtasks:

1.2.1: Create Client Server Action (1 hour) - [Backend]

- Validate user is workspace member
- Insert client record with all fields
- Return client object

1.2.2: Get All Clients Server Action (1 hour) - [Backend]

- Query all clients in workspace
- Include project count for each client
- Support basic filtering
- Return array of clients with metadata

1.2.3: Get Single Client Server Action (1 hour) - [Backend]

- Fetch client details by ID
- Include all associated projects
- Calculate project statistics
- Return enriched client object

1.2.4: Update Client Server Action (45 min) - [Backend]

- Validate user has edit permissions
- Update client fields
- Return updated client

1.2.5: Delete Client Server Action (1 hour) - [Backend]

- Validate user is admin
- Check if client has active projects
- Return warning if projects exist
- Soft delete client record

1.2.6: Test All Client Functions (15 min) - [Backend]

- Verify CRUD operations work
- Test permission enforcement

Deliverable: Complete client management logic

TASK 1.3: Project Management Logic

Goal: Build functions to manage projects **Duration:** 8 hours

Subtasks:

1.3.1: Create Project Server Action (2 hours) - [Backend]

- Validate user has create permission
- Validate client exists
- Handle due_date as date or string
- Insert project record
- Return project object with client info

1.3.2: Get All Projects Server Action (3 hours) - [Backend]

- Query all projects in workspace
- Join with clients table for client names
- Support filtering by status, client, due date range
- Support sorting by any column
- Support search by project or client name
- Return paginated results with metadata

1.3.3: Get Single Project Server Action (1 hour) - [Backend]

- Fetch project by ID
- Include client details
- Include tabs structure
- Return complete project object

1.3.4: Update Project Server Action (1 hour) - [Backend]

- Validate user has edit permissions
- Update any project field
- Handle status changes
- Return updated project

1.3.5: Delete Project Server Action (1.5 hours) - [Backend]

- Validate user is admin
- Check if project has tabs/content
- Return warning if content exists
- Cascade delete all tabs, blocks, comments, attachments

- Soft delete project record

1.3.6: Test All Project Functions (30 min) - [Backend]

- Test CRUD operations
- Test filtering and sorting
- Test cascade deletes

Deliverable: Complete project management logic

TASK 1.4: Dashboard Layout & Structure

Goal: Build the main dashboard shell with sidebar navigation **Duration:** 6 hours

Subtasks:

1.4.1: Create Dashboard Layout Component (2 hours) - [UI]

- Build sidebar with navigation links
- Sidebar sections: Projects, Clients, Payments
- Workspace switcher
- Header with user menu
- Main content area
- Responsive sidebar

1.4.2: Create Dashboard Route (1 hour) - [Backend] + [UI]

- [Backend] Create /dashboard route (30 min)
- [Backend] Fetch user's workspaces in loader (15 min)
- [UI] Redirect to /dashboard/projects (15 min)

1.4.3: Build Workspace Switcher (1.5 hours) - [UI]

- Display current workspace name
- Dropdown to switch workspaces
- Update context when switching
- Show member role badge

1.4.4: Build User Menu (1 hour) - [UI] + [Backend]

- [UI] User avatar/name in header (30 min)
- [UI] Dropdown menu (15 min)
- [Backend] Implement logout functionality (15 min)

1.4.5: Add Loading States (30 min) - [UI]

- Loading skeleton for sidebar
- Loading skeleton for main content

Deliverable: Dashboard shell with navigation

TASK 1.5: Projects Table UI

Goal: Build the main projects table view **Duration:** 8 hours

Subtasks:

1.5.1: Create Projects Table Route (1 hour) - [Backend] + [UI]

- [Backend] Create /dashboard/projects route (30 min)
- [Backend] Fetch projects using listProjects action (30 min)

1.5.2: Build Table Component (2 hours) - [UI]

- Table with columns: Client, Project, Status, Due Date
- Display project data from loader
- Proper styling with shadcn table
- Click row to open project

1.5.3: Add Status Badges (30 min) - [UI]

- Color-coded status badges
- not_started: gray, in_progress: blue, complete: green

1.5.4: Add Due Date Formatting (30 min) - [UI]

- Format dates nicely
- Show "quarterly" or "weekly" as text
- Highlight overdue dates in red

1.5.5: Add Empty State (30 min) - [UI]

- "No projects yet" message
- Button to create first project
- Helpful illustration

1.5.6: Add Loading State (30 min) - [UI]

- Skeleton table while loading
- Proper number of skeleton rows

1.5.7: Make Rows Clickable (1 hour) - [UI]

- Click row to navigate to project detail
- Hover effects
- Cursor pointer

1.5.8: Add Row Actions Menu (1.5 hours) - [UI]

- Three-dot menu on each row
- Options: Edit, Delete
- Show based on permissions

Deliverable: Working projects table display

TASK 1.6: Create Project Flow

Goal: Build UI and logic to create new projects **Duration:** 4 hours

Subtasks:

1.6.1: Build Create Project Button (15 min) - [UI]

- "+ New Project" button above table
- Opens dialog on click

1.6.2: Build Create Project Dialog (2 hours) - [UI]

- Form with fields: name, client, status, due date
- Validation messages
- Cancel and Create buttons
- Use shadcn dialog component

1.6.3: Connect to Server Action (1 hour) - [Backend] + [UI]

- [UI] Form submission (30 min)
- [Backend] Call createProject action (30 min)
- Handle success/errors

1.6.4: Add Optimistic UI (45 min) - [UI]

- Immediately add project to table

- Update with real data when server responds
- Remove if creation fails

Deliverable: Can create new projects from UI

TASK 1.7: Edit Project Flow

Goal: Build UI to edit existing projects **Duration:** 3 hours

Subtasks:

1.7.1: Build Edit Project Dialog (1.5 hours) - [UI]

- Similar to create dialog
- Pre-fill with existing project data
- Same fields as create
- Save and Cancel buttons

1.7.2: Trigger Edit from Table (30 min) - [UI]

- Click "Edit" in row actions menu
- Open edit dialog with project data loaded

1.7.3: Connect to Server Action (45 min) - [Backend] + [UI]

- [UI] Form submission (25 min)
- [Backend] Call updateProject action (20 min)
- Handle success/errors

1.7.4: Add Optimistic UI (15 min) - [UI]

- Immediately update table row
- Revert if update fails

Deliverable: Can edit projects from table

TASK 1.8: Delete Project Flow

Goal: Build UI to delete projects **Duration:** 2 hours

Subtasks:

1.8.1: Build Delete Confirmation Dialog (1 hour) - [UI]

- Warning message about deletion
- Show if project has content
- Confirm and Cancel buttons
- Use shadcn alert-dialog

1.8.2: Trigger Delete from Table (15 min) - [UI]

- Click "Delete" in row actions menu
- Only show for admins
- Open confirmation dialog

1.8.3: Connect to Server Action (30 min) - [Backend] + [UI]

- [UI] Confirmation triggers (15 min)
- [Backend] Call deleteProject action (15 min)
- Handle success/errors

1.8.4: Add Optimistic UI (15 min) - [UI]

- Immediately remove row from table
- Add back if deletion fails

Deliverable: Can delete projects with confirmation

TASK 1.9: Table Filtering & Sorting

Goal: Add ability to filter and sort projects table **Duration:** 5 hours

Subtasks:

1.9.1: Build Status Filter (1 hour) - [UI] + [Backend]

- [UI] Dropdown above table (30 min)
- [Backend] Update URL query param and refetch (30 min)

1.9.2: Build Client Filter (1 hour) - [UI] + [Backend]

- [UI] Dropdown to filter by client (30 min)
- [Backend] Update URL query param and refetch (30 min)

1.9.3: Build Search Bar (1.5 hours) - [UI] + [Backend]

- [UI] Text input above table (45 min)
- [Backend] Debounced search, updates URL param (45 min)

1.9.4: Add Sort by Column (1 hour) - [UI] + [Backend]

- [UI] Click column headers to sort (30 min)
- [Backend] Toggle ascending/descending, update URL (30 min)

1.9.5: Handle Multiple Filters (30 min) - [UI]

- Can combine status + client + search
- Clear filters button
- Show active filter count

Deliverable: Full filtering and sorting on projects table

PHASE 2: TABS & PAGES STRUCTURE

Duration: Week 3 (38-43 hours) **Goal:** Build tab system with nesting and basic page structure

---

TASK 2.1: Tab Management Logic

Goal: Build functions to manage tabs and sub-tabs **Duration:** 7 hours

Subtasks:

2.1.1: Create Tab Server Action (1.5 hours) - [Backend]

- Validate user has access to project
- Handle parent_tab_id (null for top-level)
- Calculate position
- Insert tab record
- Return tab object

2.1.2: Get Project Tabs Server Action (2 hours) - [Backend]

- Query all tabs for project
- Structure tabs in hierarchy
- Return nested structure with ordering
- Handle unlimited nesting depth

2.1.3: Update Tab Server Action (1 hour) - [Backend]

- Validate user has edit permissions
- Update tab name
- Handle moving tab to different parent
- Recalculate positions if needed

2.1.4: Reorder Tabs Server Action (1.5 hours) - [Backend]

- Accept new order array of tab IDs
- Update position field for all affected tabs
- Only reorder tabs at same level
- Validate all IDs belong to same parent

2.1.5: Delete Tab Server Action (1.5 hours) - [Backend]

- Validate user has delete permissions
- Cascade delete all sub-tabs
- Cascade delete all blocks
- Update positions of remaining tabs

Deliverable: Complete tab management logic

TASK 2.2: Project View Route

Goal: Build the project detail view with tab navigation **Duration:** 6 hours

Subtasks:

2.2.1: Create Project Detail Route (1 hour) - [Backend] + [UI]

- [Backend] Create /dashboard/projects/[projectId] route (30 min)
- [Backend] Fetch project details and tabs (30 min)

2.2.2: Build Project Header (1.5 hours) - [UI]

- Display project name prominently
- Show client name
- Show status badge
- Back button to dashboard
- Edit project button

2.2.3: Build Tab Bar Component (2 hours) - [UI]

- Display top-level tabs horizontally
- Active tab indicator
- Click to switch tabs
- Responsive (collapse to dropdown on mobile)

2.2.4: Add Tab Loading State (30 min) - [UI]

- Loading skeleton for tab bar
- Loading skeleton for page content

2.2.5: Handle No Tabs State (1 hour) - [UI]

- Empty state when project has no tabs
- Message: "Add your first tab"
- Button to create first tab

Deliverable: Project view with tab navigation

TASK 2.3: Tab Management UI

Goal: Build UI for creating, editing, and deleting tabs **Duration:** 5 hours

Subtasks:

2.3.1: Build Add Tab Button (30 min) - [UI]

- "+ Add Tab" button in tab bar
- Opens dialog to create tab

2.3.2: Build Create Tab Dialog (1.5 hours) - [UI]

- Form with tab name input
- Option to create as sub-tab
- Parent tab selector
- Create and Cancel buttons

2.3.3: Connect to Server Action (1 hour) - [Backend] + [UI]

- [UI] Form submission (30 min)
- [Backend] Call createTab action (30 min)
- Handle success/errors

2.3.4: Build Inline Tab Rename (1 hour) - [UI] + [Backend]

- [UI] Double-click tab name to edit (30 min)
- [Backend] Save on Enter or blur (30 min)

2.3.5: Build Tab Context Menu (1.5 hours) - [UI] + [Backend]

- [UI] Right-click or three-dot menu (45 min)
- [UI] Options: Rename, Add sub-tab, Delete (30 min)
- [Backend] Connect to appropriate actions (15 min)

Deliverable: Can create, rename, and delete tabs

TASK 2.4: Sub-tabs Display

Goal: Show nested sub-tabs with proper hierarchy **Duration:** 6 hours

Subtasks:

2.4.1: Build Nested Tab Renderer (2 hours) - [UI]

- Recursive component to render tab tree
- Indent sub-tabs visually
- Show nesting with icons or indentation
- Support unlimited depth

2.4.2: Add Expand/Collapse for Parent Tabs (2 hours) - [UI]

- Parent tabs can expand to show sub-tabs
- Collapsed by default
- Arrow icon indicates state
- Preserve state in localStorage or URL

2.4.3: Style Sub-tab Navigation (1 hour) - [UI]

- Different visual treatment for sub-tabs
- Smaller font or different color
- Clear parent-child relationship
- Hover effects

2.4.4: Handle Deep Nesting (1 hour) - [UI]

- Limit visual nesting to 3-4 levels
- After 3 levels, indent less
- Ensure tabs remain clickable and readable

Deliverable: Sub-tabs display with nesting

TASK 2.5: Page/Canvas Foundation

Goal: Build the page structure that holds blocks **Duration:** 6 hours

Subtasks:

2.5.1: Create Tab Page Route (1 hour) - [Backend] + [UI]

- [Backend] Create /dashboard/projects/[projectId]/tabs/[tabId] route (30 min)
- [Backend] Fetch tab details and blocks (30 min)

2.5.2: Build Page Canvas Component (1.5 hours) - [UI]

- Empty canvas container
- Proper width constraints
- Padding and spacing
- Background styling

2.5.3: Add "Add Block" Button (1 hour) - [UI]

- Button at bottom of page
- Opens block type picker
- Styled consistently

2.5.4: Build Block Type Picker (2 hours) - [UI]

- Modal or dropdown showing all block types
- Organized by category
- Icons for each type
- Description on hover

2.5.5: Add Empty Page State (30 min) - [UI]

- "No blocks yet" message
- Helpful prompt to add first block
- Large add block button

Deliverable: Page foundation ready for blocks

TASK 2.6: Basic Block Display

Goal: Display blocks on the page **Duration:** 8 hours

Subtasks:

2.6.1: Create Block Base Component (2 hours) - [UI]

- Wrapper component for all block types
- Handles block ID, type routing
- Hover effects, selection state
- Block menu trigger
- Consistent spacing

2.6.2: Build Text Block Display (1.5 hours) - [UI]

- Render plain text content
- Show as paragraph
- Handle empty text blocks
- Basic styling

2.6.3: Build Link Block Display (1.5 hours) - [UI]

- Show URL, title, description
- Card-style display
- Clickable to open in new tab
- Show favicon if available

2.6.4: Build Divider Block Display (30 min) - [UI]

- Horizontal line
- Optional label in center
- Adjustable thickness/style

2.6.5: Build Section Block Display (2 hours) - [UI]

- Container with visible border
- Can contain other blocks (except sections)
- Scrollable if content exceeds height
- Visual nesting indicator

2.6.6: Create Block Hover Menu (1.5 hours) - [UI]

- Three-dot menu appears on hover
- Menu options: Edit, Delete, Highlight, Comment, Duplicate
- Position menu in top-right of block

Deliverable: Basic blocks render and display

PHASE 3: CORE BLOCK TYPES

Duration: Week 4 (43-48 hours) **Goal:** Build all essential block types with full editing

---

TASK 3.1: Block Management Logic

Goal: Build functions to manage blocks **Duration:** 8 hours

Subtasks:

3.1.1: Create Block Server Action (2 hours) - [Backend]

- Validate user has access to tab
- Handle parent_block_id for sections
- Handle different content structures per type
- Calculate position
- Insert block record
- Return block object

3.1.2: Get Tab Blocks Server Action (2 hours) - [Backend]

- Query all blocks for tab
- Structure with nested blocks
- Order by position
- Return properly structured block tree

3.1.3: Update Block Server Action (2 hours) - [Backend]

- Validate user has edit permissions
- Handle content updates per type
- Update updated_at timestamp
- Return updated block

3.1.4: Move Block Server Action (2.5 hours) - [Backend]

- Handle moving within same tab
- Handle moving to different tab
- Handle moving in/out of sections

- Recalculate positions
- Validate can't move section into section

3.1.5: Delete Block Server Action (1.5 hours) - [Backend]

- Validate user has delete permissions
- If section, cascade delete nested blocks
- Update positions of remaining blocks
- Soft delete

Deliverable: Complete block management logic

TASK 3.2: Text Block with Rich Text

Goal: Build fully functional text editing with formatting **Duration:** 9 hours

Subtasks:

3.2.1: Build Text Editor Component (3 hours) - [UI]

- Textarea or contenteditable for input
- Click to edit, click outside to save
- Handle multi-line text
- Auto-resize height

3.2.2: Build Rich Text Toolbar (3 hours) - [UI]

- Toolbar appears above text when editing
- Buttons: Bold, Italic, Underline, Code
- Heading dropdown: Normal, H1, H2, H3
- Apply formatting to selected text
- Show active formatting state

3.2.3: Implement Auto-save (2 hours) - [Backend] + [UI]

- [UI] Debounce text changes (1 hour)
- [Backend] Save after 1 second of no typing (1 hour)
- Show saving indicator

3.2.4: Handle Markdown Shortcuts (1 hour) - [UI]

- Support markdown shortcuts

- # = H1, ## = H2

- **text** = bold, *text* = italic

Deliverable: Full-featured text block with formatting

TASK 3.3: List and Task Blocks

Goal: Build list and task functionality **Duration:** 10 hours

Subtasks:

3.3.1: Build Basic List Block (2 hours) - [UI]

- Display bullet list
- Click to edit list items
- Add new item with Enter
- Delete item with Backspace

3.3.2: Add Checklist Mode (1.5 hours) - [UI]

- Toggle between bullet and checklist
- Checkbox before each item
- Check/uncheck items
- Visual strikethrough for checked

3.3.3: Add Indent/Outdent (1.5 hours) - [UI]

- Tab key to indent item
- Shift+Tab to outdent
- Visual nesting
- Support 2-3 levels

3.3.4: Build Task Block UI (2 hours) - [UI]

- Individual task with checkbox
- Task text editable
- Check/uncheck to toggle
- Strikethrough when complete

3.3.5: Add Due Date Picker (1.5 hours) - [UI]

- Calendar picker for due date
- Display date next to task
- Highlight overdue tasks
- Clear due date option

3.3.6: Add Assignee Selector (1.5 hours) - [UI] + [Backend]

- [UI] Dropdown showing teammates and clients (1 hour)
- [Backend] Fetch workspace members (30 min)
- Show avatar/initials
- Unassign option

Deliverable: Working list and task blocks

TASK 3.4: Table Block

Goal: Build editable table block **Duration:** 8 hours

Subtasks:

3.4.1: Build Table Display (2 hours) - [UI]

- Render table with rows and columns
- Initial size: 3×3
- Proper borders and spacing
- Styled header row

3.4.2: Make Cells Editable (2 hours) - [UI]

- Click cell to edit content
- Inline text editing
- Save on blur or Enter
- Tab to move to next cell

3.4.3: Add/Delete Rows (1.5 hours) - [UI]

- "+ Add row" button below table
- Delete row button on hover
- Cannot delete if only 1 row

3.4.4: Add/Delete Columns (1.5 hours) - [UI]

- "+ Add column" button at right
- Delete column button on hover

- Cannot delete if only 1 column

3.4.5: Resize Columns (1 hour) - [UI]

- Drag column border to resize
- Visual indicator while dragging
- Maintain proportions

Deliverable: Full-featured table block

TASK 3.5: Timeline Block

Goal: Build timeline for project milestones **Duration:** 7 hours

Subtasks:

3.5.1: Build Timeline Display (2 hours) - [UI]

- Vertical timeline with events
- Date markers
- Event titles and descriptions
- Visual timeline line

3.5.2: Add Event Creation (2 hours) - [UI] + [Backend]

- [UI] "+ Add event" button and dialog (1 hour)
- [Backend] Save event to block content (1 hour)
- Events automatically sorted by date

3.5.3: Edit Events (1.5 hours) - [UI] + [Backend]

- [UI] Click event to edit (45 min)
- [Backend] Update title, date, description (45 min)

3.5.4: Delete Events (1 hour) - [UI] + [Backend]

- [UI] Delete button on each event (30 min)
- [Backend] Remove from timeline (30 min)

3.5.5: Style Timeline (30 min) - [UI]

- Different colors for past/future
- Icons for milestone types
- Responsive layout

Deliverable: Working timeline block

TASK 3.6: Stripe Payment Block

Goal: Add Stripe payment link blocks **Duration:** 3 hours

Subtasks:

3.6.1: Build Payment Link Input (1 hour) - [UI] + [Backend]

- [UI] Input field for Stripe URL (30 min)
- [Backend] Validate URL is from Stripe (30 min)

3.6.2: Display Payment Link (1 hour) - [UI]

- Show as button or embedded checkout
- "Pay Now" button
- Show payment amount if detectable

3.6.3: Handle Invalid Links (1 hour) - [UI]

- Error message for invalid URLs
- Fallback to showing link

Deliverable: Stripe payment link blocks work

TASK 3.7: Drag and Drop

Goal: Add drag-and-drop block reordering **Duration:** 6 hours

Subtasks:

3.7.1: Install dnd-kit Library (30 min) - [UI]

- Install @dnd-kit/core and @dnd-kit/sortable
- Set up basic drag context

3.7.2: Add Drag Handles to Blocks (1.5 hours) - [UI]

- Six-dot drag handle icon
- Appears on left side on hover

- Cursor changes to grab

3.7.3: Implement Drag to Reorder (2.5 hours) - [UI]

- Blocks can be dragged up/down
- Visual placeholder shows drop position
- Other blocks shift
- Smooth animations

3.7.4: Connect to Server Action (1 hour) - [Backend] + [UI]

- [UI] On drop, trigger save (30 min)
- [Backend] Call moveBlock action (30 min)
- Optimistic UI update

3.7.5: Handle Section Block Dragging (30 min) - [UI]

- Can drag blocks into/out of sections
- Visual indicator when over section
- Cannot nest sections

Deliverable: Drag-and-drop block reordering works

PHASE 4: MEDIA & EMBEDS

Duration: Week 5 (38-43 hours) **Goal:** Handle file uploads, embeds, and media blocks

---

TASK 4.1: File Storage Setup

Goal: Configure Supabase Storage for file uploads **Duration:** 3 hours

Subtasks:

4.1.1: Create Storage Bucket (30 min) - [Supabase]

- Create "project-files" bucket
- Set as public or private
- Configure size limits

4.1.2: Configure Storage RLS (1.5 hours) - [Supabase]

- Users can upload to their workspace folder
- Users can read files in their workspace
- Users can delete their own uploads
- Admins can delete any files

4.1.3: Test Upload/Download (1 hour) - [Supabase]

- Manually test upload through dashboard
- Test download via URL
- Verify permissions work

Deliverable: Supabase Storage ready for files

TASK 4.2: File Upload Logic

Goal: Build functions to handle file uploads **Duration:** 4 hours

Subtasks:

4.2.1: Create Upload File API Route (2 hours) - [Backend]

- API route to handle multipart/form-data
- Accept file from client
- Upload to Supabase Storage
- Return file URL and metadata

4.2.2: Create File Record Server Action (1 hour) - [Backend]

- After upload, create file record in database
- Store metadata: name, size, type, path
- Link to workspace
- Return file object

4.2.3: Create Attach File Server Action (30 min) - [Backend]

- Link file to specific block
- Create file_attachment record
- Set display mode (inline or linked)

4.2.4: Create Delete File Server Action (30 min) - [Backend]

- Delete from Supabase Storage
- Delete database record
- Delete attachment records

Deliverable: File upload system functional

TASK 4.3: File Upload UI

Goal: Build file upload interface **Duration:** 6 hours

Subtasks:

4.3.1: Build Drag-and-Drop Zone (2 hours) - [UI]

- Dashed border box
- "Drop files here or click to browse"
- Drag over highlights zone
- Click opens file picker

4.3.2: Add Upload Progress (1.5 hours) - [UI]

- Progress bar during upload
- Show percentage
- Show file name
- Cancel upload button

4.3.3: Show Thumbnail Preview (1 hour) - [UI]

- Preview image thumbnails during upload
- Show file icon for non-images
- Show file size

4.3.4: Handle Upload Errors (1 hour) - [UI]

- File too large error
- Unsupported file type error
- Network error handling
- Retry button

4.3.5: Support Multiple Files (30 min) - [UI]

- Can select multiple files at once
- Upload in sequence or parallel
- Show progress for each

Deliverable: File upload UI complete

TASK 4.4: Image Block

Goal: Build image display and upload **Duration:** 7 hours

Subtasks:

4.4.1: Build Image Upload (2 hours) - [UI] + [Backend]

- [UI] Click to upload or drag image (1 hour)
- [Backend] Upload to Supabase Storage (1 hour)
- Create image block with file URL

4.4.2: Display Image (1.5 hours) - [UI]

- Show image at appropriate size
- Responsive width
- Maintain aspect ratio
- Loading state

4.4.3: Add Caption Field (1 hour) - [UI] + [Backend]

- [UI] Text input below image (30 min)
- [Backend] Auto-save caption (30 min)

4.4.4: Build Image Lightbox (1.5 hours) - [UI]

- Click image to view full size
- Modal overlay
- Close on click outside or ESC
- Zoom controls optional

4.4.5: Add Resize Handles (1 hour) - [UI]

- Drag corners to resize image
- Maintain aspect ratio
- Set max/min width
- Save size preference

Deliverable: Full-featured image blocks

TASK 4.5: Video Block

Goal: Build video upload and playback **Duration:** 6 hours

Subtasks:

4.5.1: Build Video Upload (2 hours) - [UI] + [Backend]

- [UI] Upload MP4 files (1 hour)
- [Backend] Store in Supabase Storage (1 hour)
- Create video block

4.5.2: Build Video Player (2.5 hours) - [UI]

- HTML5 video player
- Play/pause button
- Progress bar
- Volume control
- Fullscreen button

4.5.3: Add Video Thumbnail (1 hour) - [UI]

- Generate thumbnail from first frame
- Show thumbnail before playing
- Play button overlay

4.5.4: Handle Large Videos (30 min) - [UI]

- Warning for very large files
- Suggest embedding instead
- Compression options

Deliverable: Video blocks work

TASK 4.6: Embed Detection & Rendering

Goal: Auto-detect and embed external content **Duration:** 8 hours

Subtasks:

4.6.1: Build URL Parser (2.5 hours) - [Backend]

- Detect URL type from pattern matching
- Extract video/document ID from URL
- Return embed configuration for Figma, Google Docs/Sheets/Slides, YouTube, Loom, Calendly

4.6.2: Build Figma Embed (1 hour) - [Backend] + [UI]

- [Backend] Parse Figma URL (30 min)
- [UI] Render in iframe (30 min)

4.6.3: Build Google Docs/Sheets/Slides Embeds (1.5 hours) - [Backend] + [UI]

- [Backend] Parse Google URL (45 min)
- [UI] Render in iframe (45 min)

4.6.4: Build YouTube Embed (45 min) - [Backend] + [UI]

- [Backend] Parse YouTube URL (20 min)
- [UI] Render in iframe (25 min)

4.6.5: Build Loom Embed (45 min) - [Backend] + [UI]

- [Backend] Parse Loom URL (20 min)
- [UI] Render in iframe (25 min)

4.6.6: Build Calendly Embed (45 min) - [Backend] + [UI]

- [Backend] Parse Calendly URL (20 min)
- [UI] Render in iframe (25 min)

4.6.7: Generic URL Fallback (45 min) - [Backend] + [UI]

- [Backend] Try generic iframe (20 min)
- [UI] If fails, show as link (25 min)

Deliverable: All major embeds supported

TASK 4.7: Embed Block UI

Goal: Build embed block interface **Duration:** 4 hours

Subtasks:

4.7.1: Build URL Input (1 hour) - [UI]

- Text input for URL
- Paste detection
- Auto-detect embed type
- Show preview

4.7.2: Display Embedded Content (1.5 hours) - [UI]

- Render iframe for supported types
- Responsive sizing
- Loading state
- Error state if embed fails

4.7.3: Add Display Mode Toggle (1 hour) - [UI] + [Backend]

- [UI] Switch between "inline" and "linked" (30 min)
- [Backend] Save preference (30 min)

4.7.4: Add "Open in new tab" Button (30 min) - [UI]

- Button in embed block
- Opens original URL
- Icon indicating external link

Deliverable: Embed blocks functional

TASK 4.8: PDF Block

Goal: Build PDF upload and viewing **Duration:** 4 hours

Subtasks:

4.8.1: Build PDF Upload (1.5 hours) - [UI] + [Backend]

- [UI] Upload PDF files (45 min)
- [Backend] Store in Supabase Storage (45 min)

4.8.2: Build PDF Viewer (2 hours) - [UI]

- Embedded PDF viewer (browser native)
- Fallback for browsers without support
- Pagination controls
- Zoom controls

4.8.3: Add Download Button (30 min) - [UI]

- "Download PDF" button
- Downloads file to computer

Deliverable: PDF blocks work

PHASE 5: COLLABORATION FEATURES

Duration: Week 6 (38-43 hours) **Goal:** Real-time updates, comments, highlights, sharing

---

TASK 5.1: Real-time Infrastructure

Goal: Set up Supabase Realtime for live collaboration **Duration:** 9 hours

Subtasks:

5.1.1: Enable Realtime in Supabase (1 hour) - [Supabase]

- Enable realtime for blocks table
- Enable realtime for comments table
- Enable realtime for projects table
- Enable realtime for tabs table

5.1.2: Create Realtime Hook (3 hours) - [Backend]

- Build custom React hook for subscriptions
- Subscribe to changes on specific tables
- Filter by workspace or project
- Handle connection/disconnection

5.1.3: Handle INSERT Events (1.5 hours) - [Backend] + [UI]

- [Backend] Detect new block/comment/tab created (45 min)
- [UI] Add to local state immediately (45 min)

5.1.4: Handle UPDATE Events (1.5 hours) - [Backend] + [UI]

- [Backend] Detect block/comment/tab updated (45 min)
- [UI] Update local state (45 min)

5.1.5: Handle DELETE Events (1.5 hours) - [Backend] + [UI]

- [Backend] Detect block/comment/tab deleted (45 min)
- [UI] Remove from local state (45 min)

5.1.6: Add Reconnection Logic (30 min) - [Backend]

- Handle network disconnections
- Automatically reconnect

- Refetch data on reconnect

Deliverable: Real-time updates working

TASK 5.2: Presence System

Goal: Show who's currently viewing workspace/tab **Duration:** 7 hours

Subtasks:

5.2.1: Set up Supabase Presence (2 hours) - [Backend]

- Use Supabase Realtime Presence
- Track when users join/leave workspace
- Store: user ID, name, avatar, location

5.2.2: Broadcast User Presence (1.5 hours) - [Backend]

- When user opens workspace, broadcast join
- When user navigates to tab, update presence
- When user closes, broadcast leave

5.2.3: Display Active Users (2 hours) - [UI]

- Show avatar bubbles in header
- List of who's viewing
- Hover to see full name
- Limit to 5 visible

5.2.4: Add "Viewing" Indicator (1 hour) - [UI]

- "3 people viewing" text
- Update count in real-time
- Animate when users join/leave

5.2.5: Color-code User Cursors (30 min) - [UI]

- Optional: show cursor position
- Each user gets a color
- Show name label with cursor

Deliverable: Presence system functional

TASK 5.3: Conflict Resolution

Goal: Handle simultaneous edits gracefully **Duration:** 3 hours

Subtasks:

5.3.1: Detect Simultaneous Edits (1.5 hours) - [Backend] + [UI]

- [Backend] Track which users editing which blocks (45 min)
- [UI] Show warning if multiple editing same block (45 min)

5.3.2: Implement Last-write-wins (1 hour) - [Backend]

- If conflict occurs, last save wins
- Show conflict warning
- Option to see what was overwritten

5.3.3: Add Optimistic UI Revert (30 min) - [UI]

- If save fails, revert UI changes
- Show error message
- Allow retry

Deliverable: Conflicts handled reasonably

TASK 5.4: Comments System

Goal: Add commenting on blocks, tabs, projects **Duration:** 10 hours

Subtasks:

5.4.1: Build Comment Logic (3 hours) - [Backend]

- createComment Server Action
- updateComment Server Action
- deleteComment Server Action
- getComments Server Action

5.4.2: Build Comment Thread UI (3 hours) - [UI]

- Sidebar that slides out showing comments
- List all comments for current target
- Show user name, avatar, timestamp

- Real-time updates

5.4.3: Add Comment Button to Blocks (1 hour) - [UI]

- "Comment" option in block menu
- Opens comment sidebar
- Highlights current block

5.4.4: Build Comment Input (1.5 hours) - [UI] + [Backend]

- [UI] Text input at bottom (45 min)
- [Backend] Submit comment (45 min)

5.4.5: Enable Comment Editing/Deleting (1.5 hours) - [UI] + [Backend]

- [UI] Edit own comments inline (45 min)
- [Backend] Delete own comments (45 min)

Deliverable: Full commenting system

TASK 5.5: Block Highlights

Goal: Allow highlighting blocks with colors **Duration:** 5 hours

Subtasks:

5.5.1: Build Highlight Logic (1.5 hours) - [Backend]

- highlightBlock Server Action
- removeHighlight Server Action
- Store color choice

5.5.2: Add Highlight Option to Block Menu (1 hour) - [UI]

- "Highlight" option in block menu
- Shows color picker

5.5.3: Build Color Picker (1.5 hours) - [UI]

- Choose from 4-5 colors
- Color swatches
- Apply immediately

5.5.4: Apply Highlight Background (1 hour) - [UI]

- Add colored background to block
- Maintain readability
- Show who highlighted on hover

5.5.5: Remove Highlight Option (30 min) - [UI] + [Backend]

- [UI] "Remove highlight" in menu (15 min)
- [Backend] Clear highlight (15 min)

Deliverable: Block highlighting works

TASK 5.6: Tab Sharing Logic

Goal: Build functions to share tabs with clients **Duration:** 7 hours

Subtasks:

5.6.1: Build Share Tab Server Action (2 hours) - [Backend]

- Accept tab ID, client email, permissions
- Generate unique access token
- Optionally set password
- Create tab_share record
- Return share URL

5.6.2: Build Share Multiple Tabs Server Action (1.5 hours) - [Backend]

- Accept array of tab IDs
- Create share record for each
- All share same access token
- Return share URL

5.6.3: Build Update Share Server Action (1 hour) - [Backend]

- Change permissions
- Change password
- Update share record

5.6.4: Build Revoke Share Server Action (1 hour) - [Backend]

- Delete share record
- Invalidate access token

5.6.5: Build Get Shared Tabs Server Action (1.5 hours) - [Backend]

- Given access token, return tabs client can see
- Check password if required
- Enforce permissions

Deliverable: Tab sharing logic complete

PHASE 6: CLIENT PORTAL

Duration: Week 7 (33-38 hours) **Goal:** Build client-facing portal

---

TASK 6.1: Portal Authentication

Goal: Allow clients to access shared tabs via link **Duration:** 8 hours

Subtasks:

6.1.1: Create Portal Access Route (2 hours) - [Backend] + [UI]

- [Backend] Create /portal/[token] route (1 hour)
- [Backend] Validate access token, fetch shared tabs (1 hour)

6.1.2: Build Client Login Page (2.5 hours) - [UI] + [Backend]

- [UI] If password required, show login form (1.5 hours)
- [Backend] Validate credentials (1 hour)

6.1.3: Validate Client Access (2 hours) - [Backend]

- Check token is valid
- Check email matches share
- Verify password if required
- Create session for client

6.1.4: Handle No Password Shares (30 min) - [Backend]

- If no password, skip login
- Go straight to portal view
- Track access

6.1.5: Build Portal Session Management (1 hour) - [Backend]

- Create session cookie for client
- Expires after 7 days
- Can logout to clear session

Deliverable: Clients can access portal

TASK 6.2: Portal RLS Policies

Goal: Secure database access for client portal users **Duration:** 3 hours

Subtasks:

6.2.1: Create Client Access Policies (1.5 hours) - [Supabase]

- Clients can SELECT tabs shared with them
- Clients can SELECT blocks in shared tabs
- Based on access_token validation

6.2.2: Create Client Comment Policies (1 hour) - [Supabase]

- If permission is "comment" or "edit"
- Clients can INSERT comments
- Clients can UPDATE/DELETE own comments

6.2.3: Create Client Edit Policies (30 min) - [Supabase]

- If permission is "edit"
- Clients can UPDATE specific blocks
- Cannot DELETE or reorder

Deliverable: Portal access secured at database level

TASK 6.3: Portal Interface

Goal: Build clean, professional client-facing UI **Duration:** 10 hours

Subtasks:

6.3.1: Build Portal Layout (2.5 hours) - [UI]

- Clean header with workspace name

- Simplified sidebar with shared tabs
- No team-only features visible
- Professional branding

6.3.2: Display Shared Tabs (2 hours) - [UI]

- List tabs client has access to
- Navigate between tabs
- Show tab hierarchy if sub-tabs shared

6.3.3: Display Blocks (Read-only mode) (2.5 hours) - [UI]

- Show all blocks in tab
- No editing UI (unless permission = edit)
- No drag handles
- Clean, distraction-free

6.3.4: Style for Clients (2 hours) - [UI]

- Larger fonts
- More whitespace
- Professional color scheme
- Print-friendly layout

6.3.5: Add "Powered by Track" Badge (1 hour) - [UI]

- Subtle badge at bottom
- Links to Track homepage

Deliverable: Client portal interface complete

TASK 6.4: Client Permissions

Goal: Implement view, comment, and edit permissions **Duration:** 7 hours

Subtasks:

6.4.1: Implement View-only Permission (1.5 hours) - [UI]

- Clients can see all content
- Cannot edit anything
- Can download files
- Can view embeds

6.4.2: Implement Comment Permission (2.5 hours) - [UI] + [Backend]

- [UI] All view-only abilities (1 hour)
- [UI] Can add comments (1 hour)
- [Backend] Comments tagged as from client (30 min)

6.4.3: Implement Edit Permission (2 hours) - [UI] + [Backend]

- [UI] All comment abilities (1 hour)
- [UI] Can edit specific blocks (1 hour)

6.4.4: Add Permission Indicators (1 hour) - [UI]

- Show what client can do
- "You can view and comment" message
- Help text if needed

Deliverable: All permissions work correctly

TASK 6.5: Sharing UI

Goal: Build interface for team to share tabs **Duration:** 8 hours

Subtasks:

6.5.1: Build Share Button (30 min) - [UI]

- "Share" button in tab header
- Opens share dialog

6.5.2: Build Share Dialog (3 hours) - [UI]

- Select tabs to share (multiselect)
- Enter client email
- Choose permissions dropdown
- Optional password input
- Generate link button

6.5.3: Generate and Display Link (1 hour) - [Backend] + [UI]

- [Backend] Call shareTab(s) action (30 min)
- [UI] Display portal URL, copy button (30 min)

6.5.4: Show Current Shares (2 hours) - [UI]

- List who has access to this tab
- Show permission level
- Edit and Revoke buttons

6.5.5: Send Share Email (1.5 hours) - [Backend]

- Integrate email service
- Send email to client with portal link
- Professional email template

Deliverable: Sharing workflow complete

PHASE 7: CLIENT & PAYMENTS PAGES

Duration: Week 8 (27-32 hours) **Goal:** Build client directory and payments tracking

---

TASK 7.1: Client Directory

Goal: Build page to manage all clients **Duration:** 12 hours

Subtasks:

7.1.1: Create Clients Route (1 hour) - [Backend] + [UI]

- [Backend] Create /dashboard/clients route (30 min)
- [Backend] Fetch all clients with project counts (30 min)

7.1.2: Build Clients List View (2.5 hours) - [UI]

- Grid or list of client cards
- Show: name, company, email, project count
- Click card to view details
- Alphabetical sorting

7.1.3: Add Client Search (1.5 hours) - [UI] + [Backend]

- [UI] Search bar above list (45 min)
- [Backend] Filter by name, company, email (45 min)

7.1.4: Build Create Client Button and Form (2 hours) - [UI] + [Backend]

- [UI] "+ New Client" button and dialog (1 hour)
- [Backend] Create client action (1 hour)

7.1.5: Create Client Detail Route (1 hour) - [Backend] + [UI]

- [Backend] Create /dashboard/clients/[clientId] route (30 min)
- [Backend] Fetch client with all projects (30 min)

7.1.6: Build Client Detail View (3 hours) - [UI]

- Display all client information
- Edit button for each field
- Notes section
- List of projects for this client

7.1.7: Enable Inline Editing (1 hour) - [UI] + [Backend]

- [UI] Click field to edit (30 min)
- [Backend] Save automatically (30 min)

Deliverable: Client directory functional

TASK 7.2: Payments System

Goal: Build page to track payments **Duration:** 10 hours

Subtasks:

7.2.1: Build Payment Logic (3 hours) - [Backend]

- createPayment Server Action
- updatePayment Server Action
- deletePayment Server Action
- listPayments Server Action

7.2.2: Create Payments Route (1 hour) - [Backend] + [UI]

- [Backend] Create /dashboard/payments route (30 min)
- [Backend] Fetch all payments (30 min)

7.2.3: Build Payments Table (2.5 hours) - [UI]

- Table: Client, Project, Amount, Status, Due Date
- Status badges

- Click row to edit
- Sort by column

7.2.4: Add Payment Filters (1.5 hours) - [UI] + [Backend]

- [UI] Filter by status, client, date range (1 hour)
- [Backend] Apply filters (30 min)

7.2.5: Build Create Payment Form (2 hours) - [UI] + [Backend]

- [UI] "+ New Payment" button and dialog (1 hour)
- [Backend] Create payment (1 hour)

7.2.6: Enable Payment Editing (1 hour) - [UI] + [Backend]

- [UI] Click row to edit (30 min)
- [Backend] Update payment (30 min)

Deliverable: Payments tracking functional

TASK 7.3: Polish & UX

Goal: Improve overall user experience **Duration:** 5 hours

Subtasks:

7.3.1: Add Loading States Everywhere (2 hours) - [UI]

- Skeleton loaders for all tables
- Spinners for form submissions
- Loading overlays

7.3.2: Add Error States (1.5 hours) - [UI]

- Error messages for failed actions
- Retry buttons
- Helpful descriptions

7.3.3: Add Empty States (1 hour) - [UI]

- Empty states for all lists
- Helpful messages
- Call-to-action buttons

7.3.4: Add Toasts for Actions (30 min) - [UI]

- Success toasts
- Error toasts
- Auto-dismiss after 3 seconds

Deliverable: Polished user experience

PHASE 8: AI INTEGRATION

Duration: Week 9 (43-48 hours) **Goal:** Add AI assistant for natural language control

---

TASK 8.1: Action Registry

Goal: Document all actions for AI to call **Duration:** 10 hours

Subtasks:

8.1.1: Create Registry File Structure (1 hour) - [Backend]

- Create /lib/ai/registry.ts
- Define registry format
- Export registry object

8.1.2: Document Workspace Actions (1 hour) - [Backend]

- createWorkspace, inviteToWorkspace, getUserWorkspaces

8.1.3: Document Project Actions (1.5 hours) - [Backend]

- createProject, updateProject, deleteProject, listProjects

8.1.4: Document Tab Actions (1.5 hours) - [Backend]

- createTab, updateTab, deleteTab, reorderTabs

8.1.5: Document Block Actions (2 hours) - [Backend]

- createBlock, updateBlock, deleteBlock, moveBlock, duplicateBlock

8.1.6: Document Content Actions (1.5 hours) - [Backend]

- formatText, createTask, completeTask, highlightBlock, createComment

8.1.7: Document Sharing Actions (1 hour) - [Backend]

- shareTab, shareTabs, revokeTabShare

8.1.8: Add Natural Language Examples (1.5 hours) - [Backend]

- For each action, add 3-5 example phrases

Deliverable: Complete action registry

TASK 8.2: Context Collector

Goal: Gather information about current state for AI **Duration:** 9 hours

Subtasks:

8.2.1: Create Context Collector Function (2 hours) - [Backend]

- Build function that gathers relevant context
- Returns structured context object

8.2.2: Capture User Context (1 hour) - [Backend]

- Current user ID, name, email
- User's role in workspace
- Permissions list

8.2.3: Capture Workspace Context (1.5 hours) - [Backend]

- Current workspace ID and name
- All workspace members
- Available clients

8.2.4: Capture Page Context (2 hours) - [Backend]

- Current project, tab
- Tab hierarchy
- Focused block

8.2.5: Capture Selection Context (1.5 hours) - [Backend]

- Selected text
- Selected blocks
- Cursor position

8.2.6: Capture Recent Actions (1 hour) - [Backend]

- Last 5 actions user performed
- Timestamps

Deliverable: Context collector functional

TASK 8.3: Command Palette UI

Goal: Build interface for AI commands **Duration:** 10 hours

Subtasks:

8.3.1: Build Command Palette Component (2.5 hours) - [UI]

- Modal overlay
- Search-style input box
- Results list below
- Keyboard navigation

8.3.2: Add Keyboard Shortcut (1 hour) - [UI]

- Cmd+K or Ctrl+K to open
- ESC to close
- Focus input when opened

8.3.3: Show Recent Actions (1.5 hours) - [UI]

- Display last actions
- Click to repeat
- Clear history option

8.3.4: Show Suggested Actions (2 hours) - [UI]

- Based on current context
- Contextual suggestions

8.3.5: Add Command Input (1.5 hours) - [UI]

- Type natural language command

- Show "processing" indicator
- Display AI response

8.3.6: Display Results and Confirmation (1.5 hours) - [UI]

- Show what AI understood
- "This will [action]. Continue?"
- Execute or Cancel buttons

Deliverable: Command palette UI complete

TASK 8.4: AI Provider Setup

Goal: Set up AI provider (Claude or DeepSeek) **Duration:** 2 hours

Subtasks:

8.4.1: Choose AI Provider (30 min) - [Backend]

- Decide: Claude or DeepSeek
- Create account
- Get API key

8.4.2: Install AI SDK (30 min) - [Backend]

- Install Vercel AI SDK or provider SDK
- Configure with API key
- Test connection

8.4.3: Create AI Client Helper (1 hour) - [Backend]

- Wrapper function for AI calls
- Handle API key securely
- Error handling

Deliverable: AI provider ready

TASK 8.5: Intent Interpretation

Goal: Build system to interpret commands and execute actions **Duration:** 12 hours

Subtasks:

8.5.1: Build Prompt Template (2.5 hours) - [Backend]

- System instructions for AI
- Include action registry
- Include context
- Define response format

8.5.2: Create AI Processing Route (2 hours) - [Backend]

- API route or Server Action
- Receives command + context
- Calls AI with prompt
- Returns structured intents

8.5.3: Parse AI Response (2 hours) - [Backend]

- Extract action name
- Extract parameters
- Extract confidence score
- Handle multiple actions

8.5.4: Validate Intents (1.5 hours) - [Backend]

- Check action exists
- Check params present
- Check permissions
- Return validation errors

8.5.5: Execute Actions (2.5 hours) - [Backend]

- Call appropriate Server Actions
- Pass extracted parameters
- Handle dependencies
- Return results

8.5.6: Handle Errors (1.5 hours) - [Backend]

- AI returns invalid action
- Missing parameters
- Action fails
- Show helpful error

Deliverable: AI can interpret and execute commands

TASK 8.6: Ambiguity Handling

Goal: Handle unclear or ambiguous commands **Duration:** 5 hours

Subtasks:

8.6.1: Detect Low Confidence (1.5 hours) - [Backend]

- If AI confidence < 70%, ask for clarification
- Parse clarification request

8.6.2: Build Clarification Dialog (2 hours) - [UI]

- Show AI's question
- Multiple choice buttons
- Text input for open clarification
- Send back to AI

8.6.3: Learn from Corrections (1.5 hours) - [Backend]

- If user corrects AI
- Log: original, AI's action, correct action
- Store for analytics

Deliverable: Ambiguity handled gracefully

TASK 8.7: Testing & Refinement

Goal: Test AI with various commands and improve **Duration:** 5 hours

Subtasks:

8.7.1: Test Simple Commands (1 hour) - [Backend] + [UI]

- "create a text block"
- "make this bold"
- Fix issues

8.7.2: Test Complex Commands (1.5 hours) - [Backend] + [UI]

- "create project for Acme with 3 tabs"

- "complete all tasks"
- Improve prompt

8.7.3: Test Ambiguous Commands (1 hour) - [Backend] + [UI]

- "add something here"
- "make it better"
- Ensure AI asks

8.7.4: Test Edge Cases (1 hour) - [Backend] + [UI]

- Invalid commands
- Missing data
- No permissions

8.7.5: Refine Prompts (30 min) - [Backend]

- Improve system instructions
- Add more examples
- Clarify descriptions

Deliverable: AI assistant works reliably

PHASE 9: FINAL POLISH & LAUNCH

Duration: Week 10 (32-37 hours) **Goal:** Bug fixes, performance, deployment

---

TASK 9.1: Comprehensive Testing

Goal: Test entire application systematically **Duration:** 8 hours

Subtasks:

9.1.1: Test Workspace Features (2 hours) - [UI] + [Backend]

- Create workspace
- Invite members
- Change roles
- Test permissions

9.1.2: Test Project Workflow (2 hours) - [UI] + [Backend]

- Create projects
- Edit, delete
- Test filtering/sorting

9.1.3: Test Tab and Block Features (2 hours) - [UI] + [Backend]

- Create tabs, sub-tabs
- Add all block types
- Edit, delete
- Drag-and-drop

9.1.4: Test Collaboration Features (2 hours) - [UI] + [Backend]

- Real-time updates
- Comments, highlights
- Sharing, portal access

Deliverable: Bug list created

TASK 9.2: Bug Fixing

Goal: Fix all critical and high-priority bugs **Duration:** 6 hours

Subtasks:

9.2.1: Fix Critical Bugs (3 hours) - [UI] + [Backend]

- Authentication issues
- Data loss bugs
- Permission bypass
- Crashes

9.2.2: Fix High-Priority Bugs (2 hours) - [UI] + [Backend]

- UI rendering issues
- Real-time sync problems
- File upload failures

9.2.3: Document Known Issues (1 hour) - [Backend]

- Create list of minor bugs
- Add to backlog

Deliverable: Application stable

TASK 9.3: Performance Optimization

Goal: Make Track fast and responsive **Duration:** 8 hours

Subtasks:

9.3.1: Measure Performance (1 hour) - [Backend] + [UI]

- Use Lighthouse
- Measure query times
- Identify slow operations

9.3.2: Optimize Database Queries (2 hours) - [Supabase] + [Backend]

- [Supabase] Add indexes (1 hour)
- [Backend] Combine queries (1 hour)

9.3.3: Optimize Client-Side (2 hours) - [UI]

- Lazy load components
- Virtualize long lists
- Debounce operations
- Reduce re-renders

9.3.4: Optimize Images and Assets (1 hour) - [UI]

- Compress images
- Use Next.js Image component
- Lazy load images

9.3.5: Add Caching (2 hours) - [Backend]

- Cache AI responses
- Cache frequently accessed data
- Use Next.js caching

Deliverable: Track loads and responds quickly

TASK 9.4: User Experience Improvements

Goal: Make Track intuitive and pleasant **Duration:** 9 hours

Subtasks:

9.4.1: Implement All Keyboard Shortcuts (3 hours) - [UI]

- Cmd+K, Cmd+B, Cmd+I, Cmd+Enter, Cmd+/
- Arrow keys, Tab
- Test all work

9.4.2: Create Shortcuts Menu (1 hour) - [UI]

- Accessible via Cmd+/ or help icon
- List all shortcuts
- Organized by category

9.4.3: Build Onboarding Flow (3 hours) - [UI]

- Welcome screen
- Interactive tutorial
- Dismissible tooltips

9.4.4: Add Contextual Help (1 hour) - [UI]

- Tooltips on key features
- Help icons
- Link to documentation

9.4.5: Improve Micro-interactions (1 hour) - [UI]

- Smooth animations
- Hover effects
- Click feedback

Deliverable: Track is intuitive and polished

TASK 9.5: Monitoring Setup

Goal: Set up error tracking and analytics **Duration:** 5 hours

Subtasks:

9.5.1: Set up Sentry (1.5 hours) - [Backend]

- Create account
- Install SDK
- Configure error catching
- Test reporting

9.5.2: Set up Analytics (2 hours) - [Backend]

- Choose: Posthog, Mixpanel, or Plausible
- Install SDK
- Track key events

9.5.3: Set up Performance Monitoring (1 hour) - [Backend]

- Track page load times
- Track query times
- Track API response times

9.5.4: Create Analytics Dashboard (30 min) - [Backend]

- Dashboard to view metrics
- Daily active users
- Features used
- Error rates

Deliverable: Monitoring in place

TASK 9.6: Production Deployment

Goal: Deploy Track to production **Duration:** 7 hours

Subtasks:

9.6.1: Production Build Testing (1 hour) - [Backend]

- Build for production locally
- Test production build
- Fix build errors

9.6.2: Set up Custom Domain (1 hour) - [Backend]

- Purchase domain
- Configure DNS
- Point to Railway

9.6.3: Configure SSL (30 min) - [Backend]

- Enable HTTPS
- Force redirect HTTP to HTTPS
- Verify certificate

9.6.4: Set up Automated Backups (1.5 hours) - [Supabase] + [Backend]

- [Supabase] Configure daily backups (1 hour)
- [Backend] Test restoration (30 min)

9.6.5: Security Audit (2 hours) - [Supabase] + [Backend]

- [Supabase] Review all RLS policies (1 hour)
- [Backend] Check auth flows, verify no sensitive data exposed (1 hour)

9.6.6: Final Deployment (1 hour) - [Backend]

- Deploy to production
- Run smoke tests
- Verify all features work

Deliverable: Track is live in production

TASK 9.7: Demo Account & Documentation

Goal: Prepare for users and marketing **Duration:** 3 hours

Subtasks:

9.7.1: Create Demo Workspace (2 hours) - [Supabase]

- Create demo account
- Populate with realistic data
- Make it look professional

9.7.2: Write Basic Documentation (1 hour) - [UI]

- How to create workspace
- How to create projects
- How to use blocks
- How to share with clients
- How to use AI assistant

Deliverable: Demo ready for showcasing

🎉 LAUNCH READY

After Week 10, Track is:

- ✅ Fully functional
- ✅ Tested and stable
- ✅ Performant
- ✅ Polished UX
- ✅ Monitored
- ✅ Deployed to production
- ✅ Ready for users

Success Metrics

By end of Week 10:

- Track is live and accessible
- Demo account shows full functionality
- No critical bugs
- Fast page loads (<2 seconds)
- Real-time works smoothly
- AI assistant works reliably
- Client portal is professional

By end of Month 3:

- 50+ active workspaces
- Positive user feedback
- Feature requests prioritized
- Revenue from paying customers
- Clear product-market fit