Frontend Design Document: 2D Image Creation & Sketching Platform

Index

1. Overview

This document outlines the frontend architecture for a cross-platform AI-powered 2D/3D creation platform. The application provides users with an intuitive canvas interface for creating, editing, and managing digital artwork through both manual drawing and AI assistance.

The platform features a comprehensive workspace for asset management, a powerful 2D canvas with integrated AI tools, and seamless integration with cloud storage and community sharing features.

2. Goals and Scope

Primary Goals

- **UI/UX Implementation**: Implement the wireframe designs for dashboard, workspace, and 2D canvas interfaces with responsive , modern-looking layouts
- **Authentication System**: Integrate Supabase Auth for secure user management and session handling
- **Asset Management & Supabase**: Build comprehensive asset storage, database integration, and workspace functionality
- **Cross-Platform Support**: Ensure consistent experience across mobile and tablet devices using React Native/Expo

3. Technology Stack

Core Framework

- **React Native**: Cross-platform mobile development
- **Expo**: Development platform and build tooling
- **TypeScript**: Type-safe development for new features
- **JavaScript**: Existing codebase compatibility

UI & Graphics

- **React Native Skia**: High-performance 2D graphics and sketching canvas
- **Expo Camera**: Camera integration for reference images
- **React Native Gesture Handler**: Touch and gesture recognition

State Management

- **Redux Toolkit**: Centralized state management
- **Redux Toolkit Query (RTK Query)**: API state management and caching
- **Redux Persist**: State persistence across app sessions

Backend Integration

- **Supabase**: Database, authentication, and storage
- **@supabase/supabase-js**: Supabase client library
- **Axios/Fetch**: HTTP client for BFF API communication

Development Tools

- **Expo CLI**: Development and build tooling
- **Metro**: JavaScript bundler
- **Flipper**: Debugging and development tools
- **ESLint/Prettier**: Code quality and formatting

4. Architecture Overview

High-Level Architecture

```
┌─────────────────────────────────────────────────────────────┐
│                    React Native App                         │
├─────────────────────────────────────────────────────────────┤
│  Screens          │  Components       │  State Management   │
│  - Dashboard      │  - Canvas         │  - Redux Store      │
│  - Workspace      │  - AI Tools       │  - RTK Query        │
│  - 2D Canvas      │  - Asset Cards    │  - Auth Slice       │
│  - Auth           │  - Modals         │  - Canvas Slice     │
├─────────────────────────────────────────────────────────────┤
│                    API Layer                                │
│  - BFF Client     │  - Supabase Client │  - Error Handling  │
│  - Auth Manager   │  - Storage Client  │  - Retry Logic     │
├─────────────────────────────────────────────────────────────┤
│                   External Services                         │
│  - BFF API        │  - Supabase       │  - AI Providers     │
│  - Authentication │  - Database       │  - (via BFF)        │
│  - Asset Storage  │  - File Storage   │                     │
└─────────────────────────────────────────────────────────────┘
```

Component Architecture

- **Screen-Level Components**: Top-level navigation and layout
- **Feature Components**: Canvas, workspace, AI tools
- **Shared Components**: Reusable UI elements, modals, forms
- **Service Layer**: API clients, authentication, storage management

Data Flow

1. **User Interaction** → Redux Actions
2. **Redux Actions** → State Updates + API Calls (RTK Query)
3. **API Responses** → State Updates + UI Re-render
4. **Asset Operations** → Supabase Storage + Database Updates

5. UI/UX Summary

Design Principles

- **Mobile-First**: Optimized for touch interaction and mobile screens
- **Progressive Disclosure**: Advanced features revealed contextually
- **Iterative Creation**: Non-destructive editing with version history
- **Hybrid Workflow**: Seamless integration of manual and AI-assisted creation

Key Interface Elements

- **Dashboard**: Central hub with creation options and asset previews
- **Workspace**: Grid-based asset management with sorting and filtering
- **2D Canvas**: Primary creation interface with drawing tools and AI panels
- **Modular AI Tools**: Contextual panels for different AI generation tasks

Responsive Design

- **Portrait Mobile**: Stacked layouts with collapsible panels
- **Landscape Mobile**: Side-by-side layouts with compact controls
- **Tablet**: Full-featured layouts with expanded tool panels

Detailed wireframes and visual specifications provided in wireframes.pdf

6. User Flow

Primary Creation Flow

```
Dashboard → "Sketch/Image" → 2D Canvas → Drawing/AI Tools → Save → Workspace
```

Asset Management Flow

```
Dashboard → "View All" → Workspace → Filter/Sort → Select Asset → Edit/View
```

AI Generation Flow

```
2D Canvas → Select AI Tool → Configure Parameters → Generate → Review Results → Iterate
```

Community Discovery Flow

```
Dashboard → "Explore Models" → Browse Public Assets → Select → Use as Reference
```

Detailed User Journeys

New User Creation

1. User opens app and sees dashboard
2. Clicks "Sketch / Image" to enter canvas
3. Uses drawing tools to create initial sketch
4. Discovers AI tools through numbered workflow (2-5)
5. Generates AI-enhanced variations
6. Saves final result to workspace

Returning User Workflow

1. User reviews workspace for recent creations
2. Selects existing asset for further editing
3. Applies additional AI modifications
4. Explores community assets for inspiration
5. Creates new variations based on discovered content

7. BFF API - 2D Endpoints

> **Note**: The Backend for Frontend (BFF) is currently under active development. Complete API documentation and endpoint access credentials will be provided to the development team upon project commencement. The endpoints outlined below represent the planned API structure. Further details to follow, including formal API definition.

Authentication

All BFF endpoints require user authentication:

- **JWT Token**: `Authorization: Bearer <token>` header for user authentication

The JWT tokens are issued by Supabase Auth and contain user session information. The BFF validates these tokens to ensure secure access to AI generation services and proper user isolation.

Image Generation Endpoints

Endpoint	Method	Purpose	Key Parameters	Response
`/generation/text-to-image`	POST	Generate images from text descriptions	prompt, style, model, size, quality	Generated image URL(s)
`/generation/image-to-image`	POST	Transform existing images with AI	input_image_url, prompt, style, similarity_strength	Modified image URL(s)
`/generation/remove-background`	POST	Remove backgrounds from images	input_image_url, output_format	Background-removed image URL
`/generation/image-inpaint`	POST	Fill selected areas with AI-generated content	input_image_url, mask_image_url, prompt	Inpainted image URL
`/generation/recolor`	POST	AI-powered color modification	input_image_url, object_prompt, color_prompt	Recolored image URL
`/tasks/{task_id}/status`	GET	Poll generation status for async operations	task_id, service (query param)	Status, progress, result URLs, error information

8. Data Model / Asset Management

Current Infrastructure (Implemented)

- **Authentication**: Full Supabase Auth with JWT tokens, MFA, SSO support
- **Asset Storage**: Supabase Storage buckets with file upload, URL management, and metadata tracking
- **Task Management**: AI service integration with status tracking and task_id workflows
- **User Management**: Complete user profiles and session handling
- **Direct Linking**: Simple relationships between images and models via source_image_id

Core Data Flow

1. **images**: All 2D content - uploads, AI-generated, and canvas creations (unified table)
2. **models**: 3D models
3. **Storage**: Task-based organization in makeit3d-app-assets bucket

Integration Strategy

- **Simplified 2D Pipeline**: All 2D content stored as images regardless of creation method
- **Unified Storage**: Single table and bucket location for uploads, AI-generated, and hand-drawn content
- **Type-Based Organization**: Use image_type field to distinguish creation methods
- **Task-Based Workflows**: Use task_id to group related assets across tables
- **Privacy Controls**: is_public field for opt-in public sharing of both images and models
- **Community Features**: Public asset discovery with efficient indexed queries
- **Shared Authentication**: Leverage existing Supabase Auth system
- **Multi-Asset Workspace**: Support for both 2D images and 3D sculpted models in user workspace
- **Bucket-Based Storage**: Organized file storage with automatic CDN and access controls

Detailed database schemas provided in Appendix C: Database Schema

8.1 Credit System & User Management

Credit Economics (Implemented ▯)

- **Base Rate**: 1 MakeIt3D Credit = $0.03
- **Free Tier**: 30 credits per user (automatic initialization)
- **Backend Authorization**: All credit checks performed server-side for security

Credit System Architecture

Frontend Responsibilities

- **Display**: Show current credit balance and operation costs
- **UX**: Preview costs before operations, optimistic UI updates
- **Error Handling**: Insufficient credit warnings and upgrade prompts
- **Subscription Management**: Plan comparison, upgrade flows, payment processing

Backend Responsibilities (In Development)

- **Authorization**: Credit checking and deduction before AI operations
- **Audit Trail**: Complete transaction logging for transparency
- **User Initialization**: Automatic 30-credit grant for new users
- **Cost Configuration**: Database-driven operation pricing

Subscription Payments & RevenueCat Integration

Multi-Platform Payment Challenge

Cross-platform apps face complex payment requirements:

- **Web**: Direct Stripe integration (97% revenue, 3% fees)
- **iOS**: Apple In-App Purchase required (70% revenue, 30% Apple cut)
- **Android**: Google Play Billing required (70% revenue, 30% Google cut)

RevenueCat Solution

RevenueCat provides unified subscription management across all platforms:

- **Single SDK**: Handles iOS, Android, and web payments
- **Backend Integration**: Server-side receipt validation and user management
- **Cross-Platform State**: Subscription status synchronized across devices
- **Analytics**: Revenue tracking and subscription metrics
- **A/B Testing**: Paywall optimization and pricing experiments

Implementation Strategy

1. **RevenueCat Configuration**: Set up iOS/Android/Web store connections
2. **Paywall Components**: Create upgrade flows with plan comparison
3. **Subscription State**: Sync RevenueCat status with database subscription_tier

4. **Credit Renewal**: Automated monthly credit grants based on active subscriptions
5. **Platform Parity**: Consistent pricing and features across all platforms

RevenueCat Integration Pattern

```
// Initialize RevenueCat and check subscription status
const checkSubscriptionStatus = async () => {
  const customerInfo = await Purchases.getCustomerInfo();
  const activeSubscription = customerInfo.activeSubscriptions[0];

  // Sync with backend credit system
  if (activeSubscription) {
    await bffClient.post('/subscriptions/sync', {
      revenuecatUserId: customerInfo.originalAppUserId,
      subscriptionTier: activeSubscription.productIdentifier,
      expiresAt: activeSubscription.expirationDate
    });
  }
};
```

Credit Integration Pattern

```
// Frontend: Credit-aware AI operation
const generateImage = async (prompt) => {
  const operationCost = 2; // Credits for Stability Core

  if (userCredits < operationCost) {
    showInsufficientCreditsModal();
    return;
  }

  // Optimistic UI update
  setCredits(prev => prev - operationCost);

  try {
    // Backend handles: credit check → deduct → AI call → log transaction
    const result = await bffClient.post('/generation/text-to-image', { prompt });
    setCredits(result.remaining_credits); // Update with actual remaining
  } catch (error) {
    setCredits(prev => prev + operationCost); // Revert on error
  }
};
```

Credit System Tables

- **user_credits**: Balance, subscription tier, lifetime stats
- **credit_transactions**: Audit trail (usage, purchases, grants, refunds)
- **operation_costs**: Configurable pricing per AI operation

Subscription Tiers - Example

Tier	Credits	Monthly Cost	Target Users
Free	30	$0	Casual users
Hobbyist	200	$9.99	Regular creators
Creator	1,200	$29.99	Professional users
Studio	4,500	$99.99	Teams & agencies

9. Security

Authentication & Authorization

- **Supabase Auth**: Email/password and OAuth provider integration
- **JWT Tokens**: Secure API communication with automatic refresh
- **Row Level Security (RLS)**: Database-level access control
- **API Key Management**: Secure BFF communication

Privacy Considerations

- **Asset Ownership**: Clear ownership and usage rights
- **Community Sharing**: Opt-in public asset sharing
- **Data Retention**: Configurable asset retention policies

11. Appendix A: Implementation Task List

□ Phase 1: Project Structure Setup (Priority: High)

- [ ] Create new directory structure under `src/features/`
  - [ ] `dashboard/` - Landing page with creation buttons
  - [ ] `workspace/` - Asset management interface
  - [ ] `canvas-2d/` - Complete 2D editor with AI tools
  - [ ] `sculpt-3d/` - Reorganized existing 3D editor
- [ ] Create new directory structure under `src/shared/`
- [ ] Add TypeScript configuration (`tsconfig.json`)
- [ ] Install new dependencies:
  - [ ] `react-native-skia` - Canvas rendering
  - [ ] `@supabase/supabase-js` - Database integration
  - [ ] `react-native-purchases` - RevenueCat SDK
  - [ ] Additional Redux Toolkit dependencies

□ Phase 2: Dashboard Implementation (Priority: High)

- [ ] Create `features/dashboard/screens/DashboardScreen.tsx`
- [ ] Implement creation buttons:
  - [ ] "Sketch / Image" → Navigate to Canvas2D
  - [ ] "Sculpture" → Navigate to existing ModelPicker
- [ ] Create `features/dashboard/components/assets/AssetGrid.tsx` (recent assets)
- [ ] Add public asset discovery components
- [ ] Integrate with credit balance display
- [ ] Update main navigation to start with Dashboard

□ Phase 3: Workspace Implementation (Priority: High)

- [ ] Create `features/workspace/screens/WorkspaceScreen.tsx`
- [ ] Implement sorting controls ("Sorting 1", "Sorting 2" from wireframes)
- [ ] Create upload functionality with camera/file integration
- [ ] Add asset filtering (image vs model types)
- [ ] Implement asset actions (fast click → edit, hold → info)
- [ ] Create shared AssetGrid and AssetCard components
- [ ] Integrate with Supabase storage for asset management

□ Phase 4: 2D Canvas Core (Priority: High)

- [ ] Create `features/canvas-2d/screens/Canvas2DScreen.tsx`
- [ ] Implement React Native Skia canvas:
  - [ ] `SketchCanvas.tsx` - Main drawing surface
  - [ ] Basic drawing tools (pen, pencil, brush, eraser)
  - [ ] Touch gesture handling for drawing
  - [ ] Zoom and pan functionality
- [ ] Create brush controls (size, type, color) from wireframe #5
- [ ] Implement undo/redo with canvas history
- [ ] Add image upload and camera integration
- [ ] Create image history component (Img1, Img2, Img3)

□ Phase 5: AI Tools Integration (Priority: High)

- [ ] Create AI tool components matching wireframe workflow:
  - [ ] `CreateImageTool.tsx` - Text-to-image (#2)

    - [ ] `ModifyImageTool.tsx` - Image-to-image (#3)
    - [ ] `ModifyAreaTool.tsx` - Inpainting (#4)
    - [ ] `RecolorTool.tsx` - AI recoloring (#5)
    - [ ] `RemoveBackgroundTool.tsx` - Background removal
    - [ ] `ChangeBackgroundTool.tsx` - Background replacement
- [ ] Implement BFF API integration:
    - [ ] `/generation/text-to-image` endpoint
    - [ ] `/generation/image-to-image` endpoint
    - [ ] `/generation/image-inpaint` endpoint
    - [ ] `/generation/recolor` endpoint
    - [ ] `/generation/remove-background` endpoint
- [ ] Add parameter controls (style, similarity, model selection)
- [ ] Implement async task polling with progress indicators

Phase 6: Credit System Integration (Priority: Medium)

- [ ] Create credit management components:
    - [ ] `CreditBalance.tsx` - Display in header/profile
    - [ ] `CreditCostPreview.tsx` - Show cost before operations
    - [ ] `InsufficientCreditsModal.tsx` - Upgrade prompts
- [ ] Implement RevenueCat integration:
    - [ ] Install and configure RevenueCat SDK
    - [ ] Create subscription tiers and paywall
    - [ ] Add cross-platform payment handling
- [ ] Add credit checking before AI operations
- [ ] Implement optimistic UI updates for credit deductions

Phase 7: 3D Editor Refactoring (Priority: Medium)

- [ ] Rename `features/advanced-editor/` → `features/sculpt-3d/`
- [ ] Move existing 3D components to new structure (keep all filenames)
- [ ] Move state slices to feature directory (no content changes)
- [ ] Update import paths in moved files
- [ ] Extract 3D-specific constants

Phase 8: Shared Infrastructure (Priority: Medium)

- [ ] Move shared components to `shared/` directory
- [ ] Create authentication components (Supabase Auth)
- [ ] Set up Supabase client configuration
- [ ] Create shared API clients (BFF, Supabase)
- [ ] Add TypeScript type definitions
- [ ] Update store configuration for new slices

Phase 9: Testing & Integration (Priority: High)

- [ ] Test complete user flow: Dashboard → Workspace → 2D Canvas
- [ ] Verify AI tool workflow (#2-5 from wireframes)
- [ ] Test credit system integration
- [ ] Verify 3D editor still works after refactoring
- [ ] Cross-platform testing (iOS/Android)
- [ ] Performance testing for canvas operations

Dependencies & Prerequisites

- [ ] Ensure Expo SDK compatibility with new packages
- [ ] Supabase project configuration and RLS policies
- [ ] BFF API endpoint availability and documentation
- [ ] Design assets and icons for 2D features

Integration Notes

- **Preserve Existing**: All current 3D editor functionality moves to `src/features/sculpt-3d/` unchanged
- **Feature Architecture**: New modules in `features/dashboard/`, `features/workspace/`, `features/canvas-2d/`
- **Shared Infrastructure**: Common components, services, and utilities in `shared/` directory
- **Hybrid Development**: TypeScript for new features, JavaScript preserved for existing code
- **State Management**: New slices integrate with existing store in `shared/state/`
- **Navigation**: New screens integrate with existing navigation system

12. Appendix B: Repository Structure

The project builds upon an existing React Native/Expo codebase with established patterns for Redux state management and component organization. The new 2D canvas features integrate seamlessly with the existing architecture through a feature-based organization that separates concerns while maintaining shared infrastructure.

> **File Operation Legend:**
>
> - `# KEEP SAME NAME` = Move existing file to new location without renaming (stays JavaScript)
> - `# NEW` = Create new file that doesn't exist yet (will be TypeScript)
> - `# MOVE FROM [path]` = File currently exists at specified path and will be relocated
> - `# EXTRACT FROM [file]` = Content extracted from existing file into new file

```
src/
├── features/
│   ├── sculpt-3d/                    # RENAMED from advanced-editor/
│   │   ├── screens/
│   │   │   └── AdvancedEditorScreen.js    # KEEP SAME NAME (move from features/advanced-editor/screens/)
│   │   ├── components/
│   │   │   ├── layout/
│   │   │   │   └── EditorLayout.js        # KEEP SAME NAME (move from features/advanced-editor/components/layout/)
│   │   │   ├── viewport/
│   │   │   │   ├── ModelViewer.js         # KEEP SAME NAME (move from src/components/editor/)
│   │   │   │   ├── Model3D.js             # KEEP SAME NAME (move from src/components/editor/)
│   │   │   │   └── BrushVisual.js         # KEEP SAME NAME (move from src/components/editor/)
│   │   │   └── tools/
│   │   │       ├── sculpting/
│   │   │       │   ├── ToolPalette.js     # KEEP SAME NAME (move from features/advanced-editor/components/tools/)
│   │   │       │   ├── BrushSettingsPanel.js  # KEEP SAME NAME (move from features/advanced-editor/components/tools/)
│   │   │       │   └── RemeshControls.js  # KEEP SAME NAME (move from features/advanced-editor/components/tools/)
│   │   │       ├── painting/
│   │   │       │   ├── ColorPalette.js    # KEEP SAME NAME (move from features/advanced-editor/components/tools/)
│   │   │       │   └── PaintBrushSettingsPanel.js  # KEEP SAME NAME (move from features/advanced-editor/components/tools/)
│   │   │       └── shared/
│   │   │           ├── ToolButton.js      # KEEP SAME NAME (move from features/advanced-editor/components/tools/)
│   │   │           └── SculptViewToggle.js # KEEP SAME NAME (move from features/advanced-editor/components/tools/)
│   │   ├── hooks/
│   │   │   ├── useSculptingSystem.js      # KEEP SAME NAME (move from src/hooks/)
│   │   │   └── useModelControls.js        # KEEP SAME NAME (move from src/hooks/)
│   │   ├── state/
│   │   │   ├── editorSlice.js             # KEEP SAME NAME - NO CONTENT CHANGES (move from src/state/slices/)
│   │   │   └── modelSlice.js              # KEEP SAME NAME - NO CONTENT CHANGES (move from src/state/slices/)
│   │   └── constants/
│   │       └── sculptTools.js             # EXTRACT FROM src/config/constants.js (3D-specific constants only)
│   │
│   ├── canvas-2d/                         # NEW FEATURE - all files are new
│   │   ├── screens/
│   │   │   └── Canvas2DScreen.tsx         # NEW
│   │   ├── components/
│   │   │   ├── layout/
│   │   │   │   ├── CanvasLayout.tsx       # NEW
│   │   │   │   ├── ToolbarLayout.tsx      # NEW
│   │   │   │   └── AIToolsLayout.tsx      # NEW
│   │   │   ├── canvas/
│   │   │   │   ├── SketchCanvas.tsx       # NEW - React Native Skia integration
│   │   │   │   ├── CanvasViewport.tsx     # NEW
│   │   │   │   ├── BrushVisualizer.tsx    # NEW
│   │   │   │   └── ImageHistory.tsx       # NEW - Img1, Img2, Img3 from wireframes
│   │   │   ├── tools/
│   │   │   │   ├── drawing/
│   │   │   │   │   ├── BrushControls.tsx  # NEW - Brush controls from wireframe #5
│   │   │   │   │   ├── DrawingToolPalette.tsx # NEW
│   │   │   │   │   ├── PenTool.tsx         # NEW
│   │   │   │   │   ├── PencilTool.tsx      # NEW
│   │   │   │   │   └── EraserTool.tsx      # NEW
│   │   │   │   ├── ai/
│   │   │   │   │   ├── CreateImageTool.tsx # NEW - Wireframe #2 workflow
│   │   │   │   │   ├── ModifyImageTool.tsx # NEW - Wireframe #3 workflow
│   │   │   │   │   ├── ModifyAreaTool.tsx  # NEW - Wireframe #4 workflow
│   │   │   │   │   ├── RecolorTool.tsx     # NEW - Wireframe #5 workflow
│   │   │   │   │   ├── RemoveBackgroundTool.tsx # NEW
│   │   │   │   │   ├── ChangeBackgroundTool.tsx # NEW
│   │   │   │   │   └── AIGenerationModal.tsx # NEW
│   │   │   │   ├── media/
│   │   │   │   │   ├── UploadTool.tsx      # NEW - From wireframes
│   │   │   │   │   ├── PhotoCapture.tsx    # NEW - Camera integration
│   │   │   │   │   └── AddImageTool.tsx    # NEW
│   │   │   │   └── selection/
│   │   │   │       ├── SelectionTools.tsx  # NEW
│   │   │   │       ├── LassoTool.tsx       # NEW
│   │   │   │       └── RectangleSelect.tsx # NEW
│   │   │   └── ui/
│   │   │       ├── UndoRedoControls.tsx    # NEW - From wireframes
│   │   │       ├── ZoomControls.tsx        # NEW
│   │   │       ├── ColorPicker.tsx         # NEW
│   │   │       └── CanvasGrid.tsx          # NEW
│   │   ├── hooks/
│   │   │   ├── useCanvas2D.ts             # NEW
│   │   │   ├── useDrawingEngine.ts        # NEW
│   │   │   ├── useAIGeneration.ts         # NEW
│   │   │   ├── useImageProcessing.ts      # NEW
│   │   │   ├── useCanvasHistory.ts        # NEW
│   │   │   ├── useSelectionSystem.ts      # NEW
│   │   │   └── useGestureHandling.ts      # NEW
│   │   ├── state/
│   │   │   └── canvas2dSlice.ts           # NEW - Complete 2D state slice
│   │   ├── utils/
│   │   │   ├── canvasOperations.ts        # NEW
│   │   │   ├── imageProcessing.ts         # NEW
│   │   │   ├── aiImageUtils.ts            # NEW
│   │   │   ├── skiaHelpers.ts             # NEW
│   │   │   └── selectionUtils.ts          # NEW
│   │   └── constants/
│   │       ├── canvas2dTools.ts           # NEW
│   │       ├── aiImageSettings.ts         # NEW
│   │       └── canvasSettings.ts          # NEW
│   │
│   ├── dashboard/                         # NEW FEATURE - all files are new
│   │   ├── screens/
│   │   │   └── DashboardScreen.tsx        # NEW
│   │   ├── components/
│   │   │   ├── layout/
│   │   │   │   └── DashboardLayout.tsx    # NEW
│   │   │   ├── creation/
│   │   │   │   ├── CreationButtons.tsx    # NEW - "Sketch/Image" & "Sculpture" buttons
│   │   │   │   ├── SketchButton.tsx       # NEW
│   │   │   │   └── SculptureButton.tsx    # NEW
│   │   │   ├── assets/
│   │   │   │   ├── AssetGrid.tsx          # NEW - Recent assets grid
│   │   │   │   ├── AssetCard.tsx          # NEW - Individual asset cards
│   │   │   │   └── ViewAllButton.tsx      # NEW - "View All" → Workspace
│   │   │   └── discovery/
│   │   │       ├── PublicAssets.tsx       # NEW
│   │   │       └── ExploreSection.tsx     # NEW
│   │   ├── hooks/
│   │   │   ├── useDashboard.ts            # NEW
│   │   │   ├── useRecentAssets.ts         # NEW
│   │   │   └── usePublicAssets.ts         # NEW
│   │   ├── state/
│   │   │   └── dashboardSlice.ts          # NEW
│   │   └── utils/
│   │       └── assetPreview.ts            # NEW
│   │
│   │
│   └── workspace/                         # NEW FEATURE - all files are new
│       ├── screens/
│       │   └── WorkspaceScreen.tsx        # NEW
│       ├── components/
│       │   ├── layout/
│       │   │   ├── WorkspaceLayout.tsx    # NEW
│       │   │   └── WorkspaceHeader.tsx    # NEW
│       │   ├── assets/
│       │   │   ├── AssetGrid.tsx          # NEW - Shared with dashboard
│       │   │   ├── AssetCard.tsx          # NEW - Shared with dashboard
│       │   │   ├── AssetInfoModal.tsx     # NEW
│       │   │   └── AssetActions.tsx       # NEW
│       │   ├── upload/
│       │   │   ├── UploadButton.tsx       # NEW
│       │   │   └── UploadModal.tsx        # NEW
│       │   ├── sorting/
│       │   │   ├── SortingControls.tsx    # NEW - "Sorting 1", "Sorting 2" from wireframes
│       │   │   ├── FilterControls.tsx     # NEW
│       │   │   └── SearchBar.tsx          # NEW
│       │   └── management/
│       │       ├── BulkActions.tsx        # NEW
│       │       └── AssetOrganizer.tsx     # NEW
│       ├── hooks/
│       │   ├── useWorkspace.ts            # NEW
│       │   ├── useAssetManagement.ts      # NEW
│       │   ├── useAssetFiltering.ts       # NEW
│       │   └── useAssetUpload.ts          # NEW
│       ├── state/
│       │   └── workspaceSlice.ts          # NEW
│       └── utils/
│           ├── assetFiltering.ts          # NEW
│           ├── assetSorting.ts            # NEW
│           └── uploadUtils.ts             # NEW
│
├── shared/                                # Shared infrastructure (mix of moved and new files)
│   ├── components/
│   │   ├── ui/                            # MOVE ENTIRE DIRECTORY (keep all existing filenames and JavaScript)
│   │   │   ├── buttons/                   # KEEP EXISTING STRUCTURE
│   │   │   │   ├── Button.tsx             # NEW - Enhanced version
│   │   │   │   └── IconButton.tsx         # NEW
│   │   │   ├── modals/                    # KEEP EXISTING STRUCTURE
│   │   │   │   ├── Modal.tsx              # NEW
│   │   │   │   └── ConfirmModal.tsx       # NEW
│   │   │   ├── inputs/                    # KEEP EXISTING STRUCTURE
│   │   │   │   ├── TextInput.tsx          # NEW
│   │   │   │   ├── Slider.tsx             # NEW
│   │   │   │   └── ColorPicker.tsx        # NEW
│   │   │   ├── feedback/                  # KEEP EXISTING STRUCTURE
│   │   │   │   ├── LoadingSpinner.tsx     # NEW
│   │   │   │   ├── ProgressBar.tsx        # NEW
│   │   │   │   └── ErrorBoundary.tsx      # NEW
│   │   │   └── layout/
│   │   │       ├── SafeAreaWrapper.tsx    # NEW
│   │   │       └── KeyboardAvoidingWrapper.tsx # NEW
│   │   ├── navigation/
│   │   │   ├── AppNavigator.tsx           # NEW - Enhanced version of existing
│   │   │   ├── AuthNavigator.tsx          # NEW
│   │   │   └── TabNavigator.tsx           # NEW
│   │   ├── layout/
│   │   │   ├── RootLayout.js              # KEEP SAME NAME (move from src/components/layout/)
│   │   │   ├── ScreenHeader.tsx           # NEW
│   │   │   └── ScreenContainer.tsx        # NEW
│   │   ├── auth/
│   │   │   ├── LoginForm.tsx              # NEW
│   │   │   ├── SignupForm.tsx             # NEW
│   │   │   ├── ProfileMenu.tsx            # NEW
│   │   │   └── AuthGuard.tsx              # NEW
│   │   └── credits/
│   │       ├── CreditBalance.tsx          # NEW
│   │       ├── CreditCostPreview.tsx      # NEW
│   │       ├── InsufficientCreditsModal.tsx # NEW
│   │       ├── SubscriptionTiers.tsx      # NEW
│   │       └── PaywallModal.tsx           # NEW
│   ├── hooks/
│   │   ├── useAppTheme.js                 # KEEP SAME NAME (move from src/hooks/)
│   │   ├── useResponsive.js               # KEEP SAME NAME (move from src/hooks/)
│   │   ├── useCryptoPolyfill.js           # KEEP SAME NAME (move from src/hooks/)
│   │   ├── useAuth.ts                     # NEW
│   │   ├── useSupabase.ts                 # NEW
│   │   ├── useCredits.ts                  # NEW
│   │   ├── useAsyncStorage.ts             # NEW
│   │   └── useApiClient.ts                # NEW
│   ├── state/
│   │   ├── store.js                       # KEEP SAME NAME (move from src/state/) - enhanced but stays JavaScript
│   │   ├── rootReducer.js                 # KEEP SAME NAME (move from src/state/) - enhanced but stays JavaScript
│   │   ├── appSlice.js                    # KEEP SAME NAME (move from src/state/slices/)
│   │   ├── authSlice.ts                   # NEW
│   │   ├── creditSlice.ts                 # NEW
│   │   └── assetSlice.ts                  # NEW
│   ├── services/
│   │   ├── api/
│   │   │   ├── bffClient.ts               # NEW
│   │   │   ├── supabaseClient.ts          # NEW
│   │   │   ├── authService.ts             # NEW
│   │   │   ├── storageService.ts          # NEW
│   │   │   ├── creditService.ts           # NEW
│   │   │   └── aiService.ts               # NEW
│   │   ├── cache/
│   │   │   ├── imageCache.ts              # NEW
│   │   │   └── assetCache.ts              # NEW
│   │   └── payments/
│   │       └── revenueCatService.ts       # NEW
│   ├── utils/
│   │   ├── fileHandling.ts                # NEW
│   │   ├── imageUtils.ts                  # NEW
│   │   ├── errorHandling.ts               # NEW
│   │   ├── validation.ts                  # NEW
│   │   └── formatting.ts                  # NEW
│   ├── types/                             # NEW DIRECTORY - TypeScript type definitions
│   │   ├── api.ts                         # NEW
│   │   ├── supabase.ts                    # NEW
│   │   ├── canvas.ts                      # NEW
│   │   ├── assets.ts                      # NEW
│   │   ├── credits.ts                     # NEW
│   │   └── navigation.ts                  # NEW
│   ├── constants/
│   │   ├── appConstants.js                # NEW (extract from src/config/constants.js) - stays JavaScript
│   │   ├── apiEndpoints.ts                # NEW
│   │   ├── creditRates.ts                 # NEW
│   │   └── subscriptionTiers.ts           # NEW
│   └── config/
│       ├── theme.js                       # KEEP SAME NAME (move from src/config/)
│       ├── supabase.ts                    # NEW
│       └── environment.ts                 # NEW
│
├── screens/                               # KEEP EXISTING - for backward compatibility with 3D flow
│   └── ModelPickerScreen.js               # KEEP SAME NAME (existing file, no changes)
│
└── assets/                                # KEEP EXISTING - no changes
    ├── images/
    ├── fonts/
    └── models/
```

File Operation Examples

"KEEP SAME NAME" Examples (JavaScript stays JavaScript):

```
# Before (current location)
src/components/editor/ModelViewer.js

# After (new location, same filename, same language)
src/features/sculpt-3d/components/viewport/ModelViewer.js

# Before (current location)
src/state/slices/editorSlice.js

# After (new location, same filename, same content, same language)
src/features/sculpt-3d/state/editorSlice.js
```

"NEW" Examples (TypeScript for new files):

```
# These files don't exist yet and will be created as TypeScript
src/features/canvas-2d/screens/Canvas2DScreen.tsx
src/features/dashboard/screens/DashboardScreen.tsx
src/shared/components/auth/LoginForm.tsx
```

13. Appendix C: Database Schema

Current Supabase Schema (Implemented)

Authentication (Built-in Supabase Auth)

- **auth.users**: Complete user management with email/password, OAuth providers
- **auth.sessions**: User session management for JWT tokens
- **auth.identities**: Multi-provider identity linking
- Full MFA, SSO, and SAML support already configured

Credit System (Implemented ✅)

```
 -- User credit balances and subscription management
user_credits (
  id UUID PRIMARY KEY,
  user_id UUID REFERENCES auth.users(id) UNIQUE,
  credits_balance INTEGER DEFAULT 30, -- Current available credits
  subscription_tier TEXT DEFAULT 'free', -- 'free', 'hobbyist', 'creator', 'studio'
  subscription_expires_at TIMESTAMPTZ, -- NULL for free tier
  total_credits_purchased INTEGER DEFAULT 0, -- Lifetime purchased
  total_credits_earned INTEGER DEFAULT 30, -- Lifetime earned (includes free)
  created_at TIMESTAMPTZ,
  updated_at TIMESTAMPTZ
)

-- Complete audit trail of all credit movements
credit_transactions (
  id UUID PRIMARY KEY,
  user_id UUID REFERENCES auth.users(id),
  transaction_type TEXT NOT NULL, -- 'usage', 'purchase', 'grant', 'refund'
  credits_amount INTEGER NOT NULL, -- Positive for additions, negative for usage
  operation_type TEXT, -- e.g., 'text_to_image_stability_core'
  operation_cost_usd DECIMAL(10,4), -- Actual API cost for tracking
  task_id TEXT, -- Link to specific task
  description TEXT, -- Human readable description
  metadata JSONB, -- Additional context
  created_at TIMESTAMPTZ
)

-- Configurable pricing for all AI operations
operation_costs (
  id UUID PRIMARY KEY,
  operation_key TEXT UNIQUE NOT NULL, -- e.g., 'text_to_image_stability_core'
  operation_name TEXT NOT NULL, -- Human readable name
  provider TEXT NOT NULL,
  credits_cost INTEGER NOT NULL, -- MakeIt3D credits required
  api_cost_usd DECIMAL(10,4) NOT NULL, -- Actual API cost
  is_active BOOLEAN DEFAULT true, -- Enable/disable operations
  created_at TIMESTAMPTZ,
  updated_at TIMESTAMPTZ
)
```

Unified Asset Pipeline (Implemented)

```
-- 2D Image Creation & Management (Unified table for ALL 2D content)
images (
  id UUID PRIMARY KEY,
  task_id TEXT NOT NULL,
  user_id UUID REFERENCES auth.users(id),
  image_type TEXT NOT NULL, -- 'upload', 'ai_generated', 'user_sketch'
  source_image_id UUID REFERENCES images(id), -- For AI transformations
  prompt TEXT,
  style TEXT,
  asset_url TEXT NOT NULL,
  status TEXT DEFAULT 'pending',
  ai_service_task_id TEXT,
  is_public BOOLEAN DEFAULT false, -- Privacy status for public sharing
  metadata JSONB, -- Creation context, AI parameters, original filename
  created_at TIMESTAMPTZ
)

-- 3D Model Generation & Management
models (
  id UUID PRIMARY KEY,
  task_id TEXT NOT NULL,
  user_id UUID REFERENCES auth.users(id),
  source_image_id UUID REFERENCES images(id),
  prompt TEXT,
  style TEXT,
  asset_url TEXT NOT NULL,
  status TEXT DEFAULT 'pending',
  ai_service_task_id TEXT,
  is_public BOOLEAN DEFAULT false, -- Privacy status for public sharing
  metadata JSONB,
  created_at TIMESTAMPTZ
)
```

Storage Structure (Supabase Storage)

```
makeit3d-app-assets/
  ├── images/{task_id}/       # All 2D content (uploads, AI-generated, sketches)
  │   ├── 0.png              # Primary image file
  │   ├── 1.png              # Additional generated variants
  │   └── original.jpg       # Original uploaded files
  ├── models/{task_id}/       # 3D model files
  │   ├── model.glb          # Generated 3D models
  │   └── preview.png        # Model preview images
  └── test_outputs/           # Development and testing assets
      ├── images/
      └── models/
```

Storage Path Patterns

- **Images**: `/images/{task_id}/{filename}` for all 2D content types
- **Models**: `/models/{task_id}/{filename}` for 3D generated models
- **Test Data**: `/test_outputs/{type}/{task_id}/{filename}` for development

Asset Relationships (Simplified & Efficient)

Primary Linking: Task ID

- **Shared Context**: Both images and models created in the same workflow share a `task_id`
- **Query Pattern**: Find related assets by joining on `task_id`

Direct Reference: Source Image ID

- **Models Table**: `source_image_id` field provides direct linking to source images
- **Use Case**: Quick lookup of the primary source image for a 3D model