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 State**: Supabase is already configured with authentication infrastructure and initial data models for the existing workflow. The schema supports a simplified asset pipeline with planned enhancements for 2D canvas features.

Current Asset Pipeline

```
2D Image Creation & Management
```

Existing 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
- **User Management**: Complete user profiles and session handling

Core Data Flow

1. **images**: All 2D content - uploads, AI-generated, and canvas creations (unified table)
2. **asset_history**: NEW - Version control and creation lineage

Planned 2D Canvas Enhancements

- **Canvas Creation**: Save canvas drawings directly as images
- **Version History**: Track creation lineage and AI modifications
- **Thumbnail Generation**: Automatic preview creation for workspace grid
- **Asset Categorization**: Distinguish between uploads, AI-generated, and hand-drawn images
- **Public Sharing**: Opt-in community asset sharing with privacy controls

Storage Structure (Supabase Storage Buckets)

```
/user-assets/{user_id}/
  /images/{asset_id}/              # ALL 2D content: uploads, AI-generated, and canvas drawings
  /models/{asset_id}/              # EXISTING: 3D sculpted models (from sculpting tool)
```

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
- **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

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: Foundation & Setup

- [ ] Add TypeScript configuration (`tsconfig.json`)
- [ ] Install new dependencies for 2D features
  - [ ] React Native Skia for canvas
  - [ ] Supabase JavaScript client
  - [ ] Additional Redux Toolkit Query dependencies
- [ ] Configure Supabase client and environment variables
- [ ] Add TypeScript type definitions directory (`/src/types/`)

Phase 2: Authentication Integration

- [ ] Install and configure Supabase Auth
- [ ] Create authentication components (TypeScript)
  - [ ] `LoginForm.tsx`
  - [ ] `SignupForm.tsx`
  - [ ] `ProfileMenu.tsx`
- [ ] Create auth Redux slice (`authSlice.ts`)
- [ ] Add authentication navigation flow
- [ ] Update existing navigation to handle auth state
- [ ] Create authentication service layer

Phase 3: Database & API Integration

- [ ] Create API client setup (`/src/state/api/`)
  - [ ] `bffClient.ts` - BFF API client with JWT auth
  - [ ] `supabaseClient.ts` - Supabase client configuration
  - [ ] `authApi.ts` - Authentication API calls
  - [ ] `canvasApi.ts` - 2D generation API calls
  - [ ] `assetsApi.ts` - Asset management API calls
- [ ] Create Supabase service layer (`/src/services/`)
  - [ ] `authService.ts`
  - [ ] `storageService.ts`
  - [ ] `cacheService.ts`

Phase 4: New Screen Infrastructure

- [ ] Create new screen components (TypeScript)
  - [ ] `DashboardScreen.tsx` - Main hub with create options
  - [ ] `WorkspaceScreen.tsx` - Asset management grid
  - [ ] `CanvasScreen.tsx` - 2D drawing interface
  - [ ] `AuthScreen.tsx` - Login/signup flow
- [ ] Update `AppNavigator.js` to include new screens
- [ ] Create auth-protected navigation wrapper
- [ ] Add navigation between existing 3D editor and new 2D features

Phase 5: 2D Canvas Implementation

- [ ] Create canvas components (`/src/components/canvas/`)
  - [ ] `SketchCanvas.tsx` - Main Skia drawing canvas
  - [ ] `BrushControls.tsx` - Drawing tool controls
  - [ ] `AIToolPanel.tsx` - AI generation interface
  - [ ] `ImageHistory.tsx` - Version management
- [ ] Create canvas Redux slice (`canvasSlice.ts`)
- [ ] Implement drawing functionality
  - [ ] Touch gesture handling
  - [ ] Brush tools (pen, brush, eraser)
  - [ ] Color and size controls
  - [ ] Undo/redo integration with existing `undoRedoManager.js`
- [ ] Add canvas export functionality (to images)

Phase 6: Workspace & Asset Management

- [ ] Create workspace components (`/src/components/workspace/`)
  - [ ] `AssetGrid.tsx` - Grid layout for all assets (2D + existing 3D)
  - [ ] `AssetCard.tsx` - Individual asset preview cards
  - [ ] `SortingControls.tsx` - Filtering and sorting
  - [ ] `UploadButton.tsx` - Asset upload functionality
- [ ] Create workspace Redux slice (`workspaceSlice.ts`)
- [ ] Integrate with existing model management from `modelSlice.js`
- [ ] Add asset versioning and history tracking
- [ ] Implement public/private sharing features

Phase 7: AI Tools Integration

- [ ] Create AI tool modals (`/src/components/modals/`)
  - [ ] `AIGenerationModal.tsx` - Parameter configuration
  - [ ] `AssetInfoModal.tsx` - Asset details and actions

- [ ] Implement BFF API integration for AI services
  - [ ] Text-to-image requests
  - [ ] Image-to-image transformations
  - [ ] Background removal
  - [ ] Inpainting tools
  - [ ] Recoloring functionality
- [ ] Add async task polling and progress indicators
- [ ] Create AI workflow UI components

Phase 8: Dashboard Integration

- [ ] Implement dashboard interface
  - [ ] "Sketch/Image" creation button → Canvas
  - [ ] "Sculpture" button → Existing 3D ModelPicker
  - [ ] Recent assets preview (both 2D and 3D)
  - [ ] "View All" → Workspace
  - [ ] Public asset discovery/browse
- [ ] Integrate with existing app theme from `useAppTheme` hook
- [ ] Update app entry point to show dashboard first

Phase 9: Shared Components & Polish

- [ ] Create shared UI components (`/src/components/shared/`)
  - [ ] `Button.tsx` - Reusable button component
  - [ ] `LoadingSpinner.tsx` - Loading states
  - [ ] `ErrorBoundary.tsx` - Error handling
- [ ] Extend existing UI components in `/src/components/ui/`
- [ ] Add responsive design optimizations
- [ ] Performance optimization for image handling
- [ ] Memory management for canvas operations

Phase 10: Testing & Integration

- [ ] Extend existing Jest test setup for TypeScript
- [ ] Add component tests for new 2D features
- [ ] Integration testing between 2D and existing 3D workflows
- [ ] Cross-platform testing (iOS/Android/Web)
- [ ] Performance testing for canvas and image operations

Dependencies & Prerequisites

- [ ] Ensure Expo SDK ~50.0.0 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 in `/src/features/advanced-editor/` remains unchanged
- **Extend Redux**: New slices integrate with existing store structure and `rootReducer.js`
- **Navigation**: New screens integrate with existing `AppNavigator.js` and React Navigation setup
- **Theming**: Leverage existing `useAppTheme` hook and theme system
- **Testing**: Build upon existing Jest + Testing Library setup
- **State Management**: Work alongside existing `editorSlice.js`, `appSlice.js`, and `modelSlice.js`

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:

```
/makeit3d-frontend
├── App.js                          # Main app entry point (existing)
├── package.json                    # Dependencies (existing + new 2D packages)
├── babel.config.js                 # Babel configuration (existing)
├── app.json                        # Expo configuration (existing)
├── index.js                        # Entry point (existing)
├── /assets                         # Static assets (existing)
├── /src
```

```
│   ├── /components                # Reusable UI components
│   │   ├── /canvas                # NEW: 2D Canvas components (TypeScript)
│   │   │   ├── SketchCanvas.tsx    # Main drawing canvas with Skia
│   │   │   ├── AIToolPanel.tsx     # AI generation tool interface
│   │   │   ├── BrushControls.tsx   # Drawing tool controls
│   │   │   └── ImageHistory.tsx    # Asset version management
│   │   ├── /workspace             # NEW: Workspace components (TypeScript)
│   │   │   ├── AssetGrid.tsx       # Grid layout for asset display
│   │   │   ├── AssetCard.tsx       # Individual asset preview cards
│   │   │   ├── SortingControls.tsx # Filtering and sorting interface
│   │   │   └── UploadButton.tsx    # Asset upload functionality
│   │   ├── /modals                # NEW: Modal components (TypeScript)
│   │   │   ├── AIGenerationModal.tsx # AI tool configuration modals
│   │   │   └── AssetInfoModal.tsx   # Asset details and actions
│   │   ├── /auth                  # NEW: Authentication components (TypeScript)
│   │   │   ├── LoginForm.tsx       # Supabase auth login
│   │   │   ├── SignupForm.tsx      # User registration
│   │   │   └── ProfileMenu.tsx     # User profile management
│   │   ├── /shared                # NEW: Shared UI components (TypeScript)
│   │   │   ├── Button.tsx          # Reusable button component
│   │   │   ├── LoadingSpinner.tsx  # Loading states
│   │   │   └── ErrorBoundary.tsx   # Error handling
│   │   └── /editor                # EXISTING: 3D sculpting components (JavaScript)
│   │       ├── BrushVisual.js      # 3D brush visualization
│   │       ├── Model3D.js          # 3D model rendering
│   │       └── ModelViewer.js      # 3D model viewer
│   ├── /screens                   # Top-level screen components
│   │   ├── DashboardScreen.tsx     # NEW: Main dashboard (TypeScript)
│   │   ├── WorkspaceScreen.tsx     # NEW: Asset management (TypeScript)
│   │   ├── CanvasScreen.tsx        # NEW: 2D drawing interface (TypeScript)
│   │   ├── AuthScreen.tsx          # NEW: Authentication flow (TypeScript)
│   │   └── ModelPickerScreen.js    # EXISTING: 3D model selection (JavaScript)
│   ├── /navigation                # Navigation configuration
│   │   ├── AppNavigator.js         # UPDATED: Main navigation (existing)
│   │   └── AuthNavigator.tsx       # NEW: Auth flow navigation (TypeScript)
│   ├── /state                     # Redux state management
│   │   ├── store.ts                # UPDATED: Redux store config (TypeScript)
│   │   ├── /slices                # Redux state slices
│   │   │   ├── authSlice.ts        # NEW: Authentication state (TypeScript)
│   │   │   ├── canvasSlice.ts      # NEW: 2D canvas state (TypeScript)
│   │   │   ├── workspaceSlice.ts   # NEW: Asset management state (TypeScript)
│   │   │   ├── appSlice.js         # EXISTING: Global app state (JavaScript)
│   │   │   ├── editorSlice.js      # EXISTING: 3D sculpting state (JavaScript)
│   │   │   └── modelSlice.js       # EXISTING: 3D model state (JavaScript)
│   │   └── /api                   # NEW: API integration (TypeScript)
│   │       ├── bffClient.ts        # BFF API client configuration
│   │       ├── supabaseClient.ts   # Supabase client setup
│   │       ├── authApi.ts          # Authentication API calls
│   │       ├── canvasApi.ts        # 2D generation API calls
│   │       └── assetsApi.ts        # Asset management API calls
│   ├── /services                  # NEW: Service layer (TypeScript)
│   │   ├── authService.ts          # Supabase auth integration
│   │   ├── storageService.ts       # Asset storage management
│   │   └── cacheService.ts         # Image and data caching
│   ├── /utils                     # Utility functions
│   │   ├── constants.ts            # NEW: App constants (TypeScript)
│   │   ├── helpers.ts              # NEW: Helper functions (TypeScript)
│   │   └── UndoRedoManager.js      # EXISTING: Undo/redo functionality (JavaScript)
│   ├── /types                     # NEW: TypeScript type definitions
│   │   ├── api.ts                  # BFF API response types
│   │   ├── supabase.ts             # Database schema types
│   │   ├── canvas.ts               # Canvas and drawing types
│   │   └── assets.ts               # Asset management types
│   └── /hooks                     # Custom React hooks
│       ├── useAuth.ts              # NEW: Authentication hook (TypeScript)
│       ├── useCanvas.ts            # NEW: Canvas state management (TypeScript)
```

```
│       ├── useCanvas.ts            # NEW: Canvas state management (TypeScript)
│       ├── useAssets.ts            # NEW: Asset management hook (TypeScript)
│       └── useSupabase.ts          # NEW: Supabase integration hook (TypeScript)
├── /config                         # NEW: Configuration files
│   ├── supabase.ts                 # Supabase configuration
│   └── constants.ts                # Environment variables and constants
└── tsconfig.json                   # NEW: TypeScript configuration
```

Integration Strategy

- **Hybrid Approach**: New 2D features developed in TypeScript while maintaining existing JavaScript components
- **Shared State**: Unified Redux store handles both new 2D canvas state and existing 3D editor state
- **Component Reuse**: Shared UI components bridge between new and existing features
- **Gradual Migration**: TypeScript adoption for new features with existing code compatibility

Key Architectural Decisions

- **Modular Structure**: Clear separation between 2D canvas, workspace, and authentication features
- **Type Safety**: TypeScript for all new components with proper type definitions
- **Service Layer**: Dedicated services for Supabase integration and asset management
- **API Abstraction**: Centralized API clients for BFF and Supabase communication

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

Existing Asset Pipeline (Implemented)

```
 -- 2D Image Creation & Management
images ( -- Unified table for all 2D content
  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,
  metadata JSONB, -- Creation context, AI parameters, original filename
  created_at TIMESTAMPTZ
)
```

Planned 2D Canvas Enhancements (To Be Implemented)

Asset History Table

```
 -- NEW: Version control for iterative 2D creation
asset_history (
  id UUID PRIMARY KEY,
  parent_asset_id UUID, -- Links to images
  child_asset_id UUID,
  operation_type TEXT, -- 'create', 'ai_generate', 'modify', 'inpaint', 'recolor'
  ai_provider TEXT, -- AI service identifier
  ai_parameters JSONB,
  created_at TIMESTAMPTZ
)
```

Public Sharing Table

```
-- NEW: Community features for public asset sharing
public_assets (
  id UUID PRIMARY KEY,
  asset_id UUID, -- References canvas_assets, images, etc.
  asset_type TEXT, -- 'canvas_asset', 'image'
  creator_id UUID REFERENCES auth.users(id),
  title TEXT,
  description TEXT,
  tags TEXT[],
  is_featured BOOLEAN DEFAULT false,
  published_at TIMESTAMPTZ
)
```

Storage Structure (Supabase Storage)

```
/user-assets/{user_id}/
  /images/{asset_id}/          # EXISTING: AI-generated images (renamed folder)
  /models/{asset_id}/          # EXISTING: 3D model files
  /canvas-assets/{asset_id}/   # NEW: 2D canvas creations
    - canvas_data.json         # Canvas state and drawing data
    - preview.png             # Thumbnail for grid display
    - export.png              # Final exported image
```