🎨 ROUTIX WEB FRONTEND - Complete Development Guide

📋 PROJECT BRIEF

Build a stunning, modern web application for **Routix** - an AI-powered YouTube thumbnail generation platform. The design must match the EXACT visual style shown in the reference images with glassmorphic cards, gradient backgrounds, and smooth animations.

Backend API: Already built and running at your backend URL **Design Reference**: Images provided (chat interface style with purple gradients and glassmorphism)

🎨 DESIGN SYSTEM (Critical - Must Match Reference Images)

Visual Style Analysis from Reference Images

Key Visual Elements:

1. **Gradient Backgrounds**: Soft purple-to-blue-to-pink gradients (`#E0C3FC → #8EC5FC → #FFE5E5`)
2. **Glassmorphism Cards**: Frosted glass effect with backdrop blur
3. **Neumorphism**: Subtle 3D raised elements
4. **Rounded Corners**: Everything uses `border-radius: 20px+`
5. **Soft Shadows**: Multiple layered shadows for depth
6. **Purple Accent Color**: `#8B7AFF` for interactive elements
7. **Clean Typography**: Dark navy text on light backgrounds

Color Palette (Extracted from Images)

```css
```

```css
/* Primary Gradient Background */
--gradient-bg: linear-gradient(135deg,
  #E0C3FC 0%,
  #D5E1FF 25%,
  #E8F4FF 50%,
  #FFE8F5 75%,
  #FFE5E5 100%
);

/* Brand Colors */
--routix-purple: #6B5DD3;      /* Logo & primary actions */
--routix-purple-light: #8B7AFF; /* Hover states */
--routix-blue: #7AA3FF;        /* Secondary accent */

/* Card Colors */
--card-bg: rgba(255, 255, 255, 0.8);
--card-border: rgba(255, 255, 255, 0.5);
--glass-bg: rgba(255, 255, 255, 0.4);

/* Text Colors */
--text-primary: #2D2A4A;       /* Dark navy */
--text-secondary: #6B6B8D;     /* Muted purple-gray */
--text-muted: #A0A0B8;         /* Light gray */

/* Status Colors */
--success: #5DD39E;
--warning: #FFB84D;
--error: #FF6B6B;
```

Typography

css

```
/* Font Stack */
font-family: 'SF Pro Display', -apple-system, BlinkMacSystemFont, 'Segoe UI',
        'Roboto', 'Oxygen', 'Ubuntu', sans-serif;

/* Font Sizes */
--text-xs: 12px;
--text-sm: 14px;
--text-base: 16px;
--text-lg: 18px;
--text-xl: 20px;
--text-2xl: 24px;
--text-3xl: 32px;
--text-4xl: 40px;

/* Font Weights */
--font-regular: 400;
--font-medium: 500;
--font-semibold: 600;
--font-bold: 700;
```

Spacing System (8px base)

css

```
--space-1: 8px;
--space-2: 16px;
--space-3: 24px;
--space-4: 32px;
--space-5: 40px;
--space-6: 48px;
--space-8: 64px;
```

Effects & Shadows

css

```css
/* Glassmorphism Effect */
.glass-card {
  background: rgba(255, 255, 255, 0.4);
  backdrop-filter: blur(20px) saturate(180%);
  border: 1px solid rgba(255, 255, 255, 0.5);
  box-shadow:
    0 8px 32px rgba(31, 38, 135, 0.08),
    inset 0 1px 0 rgba(255, 255, 255, 0.8);
}

/* Neumorphic Elevation */
.elevated {
  box-shadow:
    12px 12px 24px rgba(174, 174, 192, 0.2),
    -12px -12px 24px rgba(255, 255, 255, 0.8);
}

/* Button Shadow */
.button-shadow {
  box-shadow:
    0 4px 15px rgba(107, 93, 211, 0.3),
    0 2px 4px rgba(107, 93, 211, 0.2);
}
```

🛠️ TECH STACK

json

```json
{
  "framework": "Next.js 14+ (App Router)",
  "language": "TypeScript",
  "styling": "Tailwind CSS + Custom CSS for glassmorphism",
  "ui": "Custom components (match reference design)",
  "animations": "Framer Motion",
  "state": "Zustand",
  "api": "Axios + TanStack Query",
  "forms": "React Hook Form + Zod",
  "icons": "Lucide React",
  "toast": "Sonner"
}
```

📁 PROJECT STRUCTURE

```
routix-web/
├── app/
│   ├── (auth)/
│   │   ├── login/page.tsx
│   │   ├── register/page.tsx
│   │   └── layout.tsx
│   │
│   ├── (dashboard)/
│   │   ├── dashboard/page.tsx          # Main dashboard
│   │   ├── generate/page.tsx           # ⭐ MAIN FEATURE
│   │   ├── history/page.tsx
│   │   ├── assets/page.tsx
│   │   └── layout.tsx
│   │
│   ├── (admin)/
│   │   ├── admin/
│   │   │   ├── page.tsx
│   │   │   ├── templates/page.tsx      # ⭐ DRAG & DROP UPLOAD
│   │   │   ├── algorithms/page.tsx
│   │   │   ├── users/page.tsx
│   │   │   ├── generations/page.tsx
│   │   │   └── analytics/page.tsx
│   │   └── layout.tsx
│   │
│   ├── page.tsx                    # Landing page
│   └── layout.tsx
│
├── components/
│   ├── ui/                         # Base UI components
│   │   ├── GlassCard.tsx               # Glassmorphic card
│   │   ├── GradientButton.tsx          # Purple gradient button
│   │   ├── Input.tsx                   # Styled input
│   │   ├── StatusBadge.tsx             # Status indicator
│   │   └── Logo.tsx                    # Routix logo
│   │
│   ├── generation/
│   │   ├── ChatInterface.tsx           # Chat-like generation UI
│   │   ├── MessageBubble.tsx           # User/system message bubbles
│   │   ├── ProgressStatus.tsx          # "Syncing with Routix Intelligence..."
│   │   ├── AlgorithmSelector.tsx
│   │   └── ResultCard.tsx
```

```
│   │
│   ├── admin/
│   │   ├── TemplateUploader.tsx          # Drag & drop zone
│   │   ├── TemplateGrid.tsx
│   │   └── StatsCard.tsx
│   │
│   └── layout/
│       ├── Sidebar.tsx
│       └── Header.tsx
│
├── lib/
│   ├── api.ts                    # API client
│   ├── auth.ts
│   └── utils.ts
│
├── hooks/
│   ├── useGeneration.ts
│   └── usePolling.ts
│
├── store/
│   └── useStore.ts
│
├── styles/
│   └── globals.css
│
└── types/
    └── index.ts
```

🎯 CRITICAL COMPONENT: Generation Interface (Match Image 1)

This is THE MOST IMPORTANT page. It must look EXACTLY like the reference image.

File: `app/(dashboard)/generate/page.tsx`

```tsx
```

```tsx
'use client'

import { useState } from 'react'
import { motion, AnimatePresence } from 'framer-motion'
import { GlassCard } from '@/components/ui/GlassCard'
import { ChatInterface } from '@/components/generation/ChatInterface'
import { ProgressStatus } from '@/components/generation/ProgressStatus'
import { ResultCard } from '@/components/generation/ResultCard'

export default function GeneratePage() {
  const [stage, setStage] = useState<'input' | 'processing' | 'result'>('input')
  const [generationData, setGenerationData] = useState(null)

  return (
    <div className="min-h-screen relative overflow-hidden">
      {/* Gradient Background (EXACT match to image) */}
      <div className="fixed inset-0 bg-gradient-to-br from-[#E0C3FC] via-[#D5E1FF] via-[#E8F4FF] to-[#FFE5E5] -z-10" />

      {/* Animated gradient orbs (optional enhancement) */}
      <div className="fixed inset-0 -z-10 overflow-hidden">
        <motion.div
          className="absolute w-96 h-96 bg-purple-300/30 rounded-full blur-3xl"
          animate={{
            x: [0, 100, 0],
            y: [0, -100, 0],
          }}
          transition={{ duration: 20, repeat: Infinity }}
          style={{ top: '10%', left: '20%' }}
        />
        <motion.div
          className="absolute w-96 h-96 bg-blue-300/30 rounded-full blur-3xl"
          animate={{
            x: [0, -100, 0],
            y: [0, 100, 0],
          }}
          transition={{ duration: 25, repeat: Infinity }}
          style={{ bottom: '10%', right: '20%' }}
        />
      </div>

      {/* Main Content */}
      <div className="container max-w-4xl mx-auto px-4 py-8">
        <AnimatePresence mode="wait">
```

```
        {stage === 'input' && (
          <ChatInterface
            onSubmit={(data) => {
              setGenerationData(data)
              setStage('processing')
            }}
          />
        )}

        {stage === 'processing' && (
          <ProgressStatus
            generationId={generationData?.id}
            onComplete={() => setStage('result')}
          />
        )}

        {stage === 'result' && (
          <ResultCard
            result={generationData}
            onNewGeneration={() => setStage('input')}
          />
        )}
      </AnimatePresence>
    </div>
  </div>
 )
}
```

🎨 COMPONENT: Chat Interface (Match Image 2)

File: `components/generation/ChatInterface.tsx`

tsx

```tsx
'use client'

import { useState } from 'react'
import { motion } from 'framer-motion'
import { GlassCard } from '@/components/ui/GlassCard'
import { GradientButton } from '@/components/ui/GradientButton'
import { Logo } from '@/components/ui/Logo'
import { ArrowUp } from 'lucide-react'

interface ChatInterfaceProps {
  onSubmit: (data: any) => void
}

export function ChatInterface({ onSubmit }: ChatInterfaceProps) {
  const [prompt, setPrompt] = useState('')
  const [userName] = useState('Armando') // From auth context

  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault()
    if (!prompt.trim()) return

    // Call API
    const result = await generateThumbnail({ prompt })
    onSubmit(result)
  }

  return (
    <motion.div
      initial={{ opacity: 0, y: 20 }}
      animate={{ opacity: 1, y: 0 }}
      exit={{ opacity: 0, y: -20 }}
      className="space-y-8"
    >
      {/* Logo */}
      <motion.div
        className="flex justify-center"
        initial={{ scale: 0.8, opacity: 0 }}
        animate={{ scale: 1, opacity: 1 }}
        transition={{ delay: 0.2 }}
      >
        <div className="w-20 h-20 rounded-3xl bg-white/80 backdrop-blur-xl shadow-lg flex items-center justify-center">
          <Logo className="w-12 h-12" />
        </div>
```

```
</motion.div>

{/* Greeting */}
<motion.div
  initial={{ opacity: 0, y: 20 }}
  animate={{ opacity: 1, y: 0 }}
  transition={{ delay: 0.3 }}
  className="text-center space-y-2"
>
  <h1 className="text-4xl font-bold text-[#2D2A4A]">
    Hey {userName}!
  </h1>
  <h2 className="text-3xl font-semibold text-[#2D2A4A]">
    Can I help you with anything?
  </h2>
  <p className="text-[#6B6B8D] text-lg">
    Ready to assist you with anything you need.
  </p>
</motion.div>

{/* Input Form */}
<motion.div
  initial={{ opacity: 0, y: 20 }}
  animate={{ opacity: 1, y: 0 }}
  transition={{ delay: 0.4 }}
>
  <form onSubmit={handleSubmit}>
    <GlassCard className="p-2">
      <div className="flex items-center gap-2">
        <input
          type="text"
          value={prompt}
          onChange={(e) => setPrompt(e.target.value)}
          placeholder="Ask anything you need"
          className="flex-1 bg-transparent border-none outline-none px-4 py-3 text-[#2D2A4A] placeholder:text-[#A0A0B
        />

        <GradientButton
          type="submit"
          className="px-6 py-3 rounded-2xl"
          disabled={!prompt.trim()}
        >
          <span className="font-medium">Send</span>
          <ArrowUp className="w-5 h-5 ml-2" />
```

```
            </GradientButton>
          </div>
        </GlassCard>
      </form>
    </motion.div>

    {/* Quick Actions (Optional) */}
    <motion.div
      initial={{ opacity: 0 }}
      animate={{ opacity: 1 }}
      transition={{ delay: 0.5 }}
      className="flex justify-center gap-3"
    >
      <QuickActionButton icon="🎨" label="Create Thumbnail" />
      <QuickActionButton icon="💡" label="Brainstorm" />
      <QuickActionButton icon="📊" label="Analyze" />
    </motion.div>
  </motion.div>
 )
}

function QuickActionButton({ icon, label }: { icon: string; label: string }) {
 return (
  <button className="group">
    <GlassCard className="px-4 py-3 hover:scale-105 transition-transform">
      <div className="flex flex-col items-center gap-2">
        <span className="text-2xl">{icon}</span>
        <span className="text-sm text-[#6B6B8D] font-medium">
          {label}
        </span>
      </div>
    </GlassCard>
  </button>
 )
}
```

🎨 COMPONENT: Progress Status (Match Image 1 - Processing State)

File: `components/generation/ProgressStatus.tsx`

tsx

```tsx
'use client'

import { useEffect, useState } from 'react'
import { motion, AnimatePresence } from 'framer-motion'
import { GlassCard } from '@/components/ui/GlassCard'
import { Logo } from '@/components/ui/Logo'
import { useGenerationPolling } from '@/hooks/usePolling'

interface ProgressStatusProps {
  generationId: string
  onComplete: () => void
}

export function ProgressStatus({ generationId, onComplete }: ProgressStatusProps) {
  const { data, isLoading } = useGenerationPolling(generationId)

  const statusMessages = {
    analyzing: "Analyzing your request...",
    searching: "Matching with templates...",
    composing: "Composing design DNA...",
    generating: "Generating your thumbnail preview..."
  }

  useEffect(() => {
    if (data?.status === 'completed') {
      setTimeout(onComplete, 1000)
    }
  }, [data?.status])

  return (
    <motion.div
      initial={{ opacity: 0, scale: 0.9 }}
      animate={{ opacity: 1, scale: 1 }}
      exit={{ opacity: 0, scale: 0.9 }}
      className="flex flex-col items-center justify-center min-h-[600px] space-y-8"
    >
      {/* Logo */}
      <motion.div
        className="w-24 h-24 rounded-3xl bg-white/80 backdrop-blur-xl shadow-lg flex items-center justify-center"
        animate={{
          scale: [1, 1.05, 1],
        }}
        transition={{
```

```
    duration: 2,
    repeat: Infinity,
    ease: "easeInOut"
  }}
>
  <Logo className="w-16 h-16" />
</motion.div>

{/* Main Message */}
<motion.div
  initial={{ opacity: 0, y: 10 }}
  animate={{ opacity: 1, y: 0 }}
  className="text-center"
>
  <h2 className="text-2xl font-semibold text-[#2D2A4A] mb-2">
    Syncing with Routix Intelligence...
  </h2>
</motion.div>

{/* Status Card */}
<GlassCard className="w-full max-w-md p-8">
  <AnimatePresence mode="wait">
    <motion.div
      key={data?.status}
      initial={{ opacity: 0, x: -20 }}
      animate={{ opacity: 1, x: 0 }}
      exit={{ opacity: 0, x: 20 }}
      className="space-y-4"
    >
      <div className="flex items-center gap-3">
        <span className="text-2xl">👋</span>
        <h3 className="text-xl font-semibold text-[#2D2A4A]">
          Got it ROUTIN
        </h3>
      </div>

      {/* Status Steps */}
      <div className="space-y-3 text-[#2D2A4A]">
        <StatusStep
          text={statusMessages.analyzing}
          active={data?.status === 'analyzing'}
          completed={['searching', 'composing', 'generating'].includes(data?.status)}
        />
        <StatusStep
```

```
            text={statusMessages.searching}
            active={data?.status === 'searching'}
            completed={['composing', 'generating'].includes(data?.status)}
          />
          <StatusStep
            text={statusMessages.composing}
            active={data?.status === 'composing'}
            completed={data?.status === 'generating'}
          />
          <StatusStep
            text={statusMessages.generating}
            active={data?.status === 'generating'}
            completed={false}
          />
        </div>
      </motion.div>
    </AnimatePresence>
  </GlassCard>

  {/* Bottom Input (Disabled state) */}
  <GlassCard className="w-full max-w-2xl p-4 opacity-50">
    <div className="flex items-center gap-3 px-4">
      <span className="text-[#A0A0B8]">Generating response...</span>
      <motion.div
        className="ml-auto"
        animate={{ rotate: 360 }}
        transition={{ duration: 1, repeat: Infinity, ease: "linear" }}
      >
        <ArrowUp className="w-5 h-5 text-[#8B7AFF]" />
      </motion.div>
    </div>
  </GlassCard>
 </motion.div>
 )
}

function StatusStep({
 text,
 active,
 completed
}: {
 text: string
 active: boolean
 completed: boolean
```

```
}) {
  return (
    <motion.div
      className="flex items-center gap-2"
      initial={{ opacity: 0.3 }}
      animate={{
        opacity: active ? 1 : completed ? 0.6 : 0.3,
      }}
    >
      {completed ? (
        <span className="text-green-500">✓</span>
      ) : active ? (
        <motion.span
          animate={{ scale: [1, 1.2, 1] }}
          transition={{ duration: 1, repeat: Infinity }}
        >
          ⏳
        </motion.span>
      ) : (
        <span className="opacity-30">○</span>
      )}
      <span className={active ? 'font-medium' : ''}>
        {text}
      </span>
    </motion.div>
  )
}
```

🎨 BASE COMPONENT: GlassCard

File: `components/ui/GlassCard.tsx`

tsx

```tsx
import { cn } from '@/lib/utils'

interface GlassCardProps {
  children: React.ReactNode
  className?: string
  hover?: boolean
}

export function GlassCard({ children, className, hover = false }: GlassCardProps) {
  return (
    <div
      className={cn(
        // Base glass effect
        'relative',
        'bg-white/40',
        'backdrop-blur-xl',
        'backdrop-saturate-150',
        'border border-white/50',
        'rounded-3xl',
        'shadow-[0_8px_32px_rgba(31,38,135,0.08)]',

        // Inner glow
        'before:absolute before:inset-0',
        'before:rounded-3xl',
        'before:p-[1px]',
        'before:bg-gradient-to-b before:from-white/80 before:to-transparent',
        'before:-z-10',

        // Hover effect
        hover && 'transition-all duration-300 hover:scale-[1.02] hover:shadow-[0_12px_40px_rgba(31,38,135,0.12)]',

        className
      )}
    >
      {children}
    </div>
  )
}
```

🎨 BASE COMPONENT: Gradient Button

File: components/ui/GradientButton.tsx

tsx

```tsx
import { motion } from 'framer-motion'
import { cn } from '@/lib/utils'

interface GradientButtonProps extends React.ButtonHTMLAttributes<HTMLButtonElement> {
  children: React.ReactNode
  variant?: 'primary' | 'secondary'
}

export function GradientButton({
  children,
  className,
  variant = 'primary',
  ...props
}: GradientButtonProps) {
  return (
    <motion.button
      whileHover={{ scale: 1.02 }}
      whileTap={{ scale: 0.98 }}
      className={cn(
        'relative px-6 py-3 rounded-2xl',
        'font-semibold text-white',
        'shadow-[0_4px_15px_rgba(107,93,211,0.3)]',
        'transition-all duration-200',
        'disabled:opacity-50 disabled:cursor-not-allowed',

        // Gradient background
        variant === 'primary' && 'bg-gradient-to-r from-[#6B5DD3] to-[#8B7AFF]',
        variant === 'secondary' && 'bg-gradient-to-r from-[#7AA3FF] to-[#A8C5FF]',

        // Hover effect
        'hover:shadow-[0_6px_20px_rgba(107,93,211,0.4)]',

        className
      )}
      {...props}
    >
      <span className="relative z-10 flex items-center justify-center">
        {children}
      </span>
    </motion.button>
  )
}
```

🎨 ADMIN: Template Upload (Drag & Drop)

File: `app/(admin)/admin/templates/page.tsx`

```tsx
```

```tsx
'use client'

import { useState } from 'react'
import { useDropzone } from 'react-dropzone'
import { GlassCard } from '@/components/ui/GlassCard'
import { GradientButton } from '@/components/ui/GradientButton'
import { Upload, X, Loader2 } from 'lucide-react'
import { motion, AnimatePresence } from 'framer-motion'

export default function TemplatesPage() {
  const [files, setFiles] = useState<File[]>([])
  const [uploading, setUploading] = useState(false)

  const { getRootProps, getInputProps, isDragActive } = useDropzone({
    accept: { 'image/*': ['.jpg', '.jpeg', '.png', '.webp'] },
    multiple: true,
    onDrop: (acceptedFiles) => {
      setFiles(prev => [...prev, ...acceptedFiles])
    }
  })

  const handleUpload = async () => {
    setUploading(true)

    const formData = new FormData()
    files.forEach(file => formData.append('files', file))

    try {
      const response = await fetch('/api/admin/templates/bulk-upload', {
        method: 'POST',
        body: formData
      })

      const result = await response.json()
      // Show success toast
      // Trigger analysis

    } catch (error) {
      console.error(error)
    } finally {
      setUploading(false)
    }
  }
```

```
return (
  <div className="min-h-screen relative overflow-hidden">
    {/* Same gradient background */}
    <div className="fixed inset-0 bg-gradient-to-br from-[#E0C3FC] via-[#D5E1FF] to-[#FFE5E5] -z-10" />

    <div className="container max-w-6xl mx-auto px-4 py-8">
      <div className="mb-8">
        <h1 className="text-4xl font-bold text-[#2D2A4A] mb-2">
          Template Library
        </h1>
        <p className="text-[#6B6B8D] text-lg">
          Upload and manage your secret template collection
        </p>
      </div>

      {/* Drag & Drop Zone */}
      <GlassCard className="p-8 mb-6">
        <div
          {...getRootProps()}
          className={cn(
            'border-2 border-dashed rounded-2xl p-12 text-center cursor-pointer transition-all',
            isDragActive ? 'border-[#8B7AFF] bg-[#8B7AFF]/10' : 'border-[#6B6B8D]/30'
          )}
        >
          <input {...getInputProps()} />

          <motion.div
            animate={isDragActive ? { scale: 1.1 } : { scale: 1 }}
            className="space-y-4"
          >
            <div className="w-20 h-20 mx-auto rounded-full bg-[#8B7AFF]/20 flex items-center justify-center">
              <Upload className="w-10 h-10 text-[#8B7AFF]" />
            </div>

            {isDragActive ? (
              <p className="text-xl font-semibold text-[#2D2A4A]">
                Drop files here!
              </p>
            ) : (
              <>
                <h3 className="text-xl font-semibold text-[#2D2A4A]">
                  Drag & Drop Templates
                </h3>
```

```
          <p className="text-[#6B6B8D]">
            or click to browse • JPG, PNG, WebP • Max 10MB each
          </p>
        </>
      )}
    </motion.div>
  </div>
</GlassCard>

{/* File List */}
<AnimatePresence>
  {files.length > 0 && (
    <motion.div
      initial={{ opacity: 0, y: 20 }}
      animate={{ opacity: 1, y: 0 }}
      exit={{ opacity: 0, y: -20 }}
    >
      <GlassCard className="p-6 mb-6">
        <div className="flex justify-between items-center mb-4">
          <h3 className="text-lg font-semibold text-[#2D2A4A]">
            {files.length} files ready to upload
          </h3>
          <GradientButton onClick={handleUpload} disabled={uploading}>
            {uploading ? (
              <>
                <Loader2 className="w-4 h-4 animate-spin mr-2" />
                Uploading...
              </>
            ) : (
              <>
                <Upload className="w-4 h-4 mr-2" />
                Upload All
              </>
            )}
          </GradientButton>
        </div>

        <div className="grid grid-cols-4 gap-4">
          {files.map((file, index) => (
            <motion.div
              key={index}
              initial={{ opacity: 0, scale: 0.8 }}
              animate={{ opacity: 1, scale: 1 }}
              exit={{ opacity: 0, scale: 0.8 }}
```

```
                  className="relative group"
                >
                  <div className="aspect-video rounded-xl overflow-hidden bg-white/60">
                    <img
                      src={URL.createObjectURL(file)}
                      alt={file.name}
                      className="w-full h-full object-cover"
                    />
                  </div>

                  <button
                    onClick={() => setFiles(files.filter((_, i) => i !== index))}
                    className="absolute -top-2 -right-2 w-8 h-8 rounded-full bg-red-500 text-white flex items-center justify-cente
                  >
                    <X className="w-4 h-4" />
                  </button>

                  <p className="mt-2 text-sm text-[#6B6B8D] truncate">
                    {file.name}
                  </p>
                  <p className="text-xs text-[#A0A0B8]">
                    {(file.size / 1024 / 1024).toFixed(2)} MB
                  </p>
                </motion.div>
              ))}
            </div>
          </GlassCard>
        </motion.div>
      )}
    </AnimatePresence>

    {/* Template Grid */}
    <div className="grid grid-cols-4 gap-4">
      {/* Existing templates... */}
    </div>
  </div>
</div>
)
}
```

GLOBAL STYLES

File: styles/globals.css

```css
```

```
@tailwind base;
@tailwind components;
@tailwind utilities;

/* Font Import */
@import url('https://fonts.googleapis.com/css2?family=Inter:wght@400;500;600;700;800&display=swap');

:root {
  /* Colors */
  --routix-purple: #6B5DD3;
  --routix-purple-light: #8B7AFF;
  --routix-blue: #7AA3FF;

  --text-primary: #2D2A4A;
  --text-secondary: #6B6B8D;
  --text-muted: #A0A0B8;
}

* {
  box-sizing: border-box;
  padding: 0;
  margin: 0;
}

html,
body {
  font-family: 'Inter', -apple-system, BlinkMacSystemFont, 'Segoe UI', sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

/* Custom Scrollbar */
::-webkit-scrollbar {
  width: 10px;
}

::-webkit-scrollbar-track {
  background: rgba(255, 255, 255, 0.1);
  border-radius: 10px;
}

::-webkit-scrollbar-thumb {
  background: rgba(107, 93, 211, 0.4);
```

```
  border-radius: 10px;
}

::-webkit-scrollbar-thumb:hover {
  background: rgba(107, 93, 211, 0.6);
}

/* Glassmorphism utility classes */
.glass {
  background: rgba(255, 255, 255, 0.4);
  backdrop-filter: blur(20px) saturate(180%);
  -webkit-backdrop-filter: blur(20px) saturate(180%);
}

.glass-dark {
  background: rgba(255, 255, 255, 0.2);
  backdrop-filter: blur(20px) saturate(180%);
  -webkit-backdrop-filter: blur(20px) saturate(180%);
}

/* Gradient text */
.gradient-text {
  background: linear-gradient(135deg, #6B5DD3 0%, #8B7AFF 100%);
  -webkit-background-clip: text;
  -webkit-text-fill-color: transparent;
  background-clip: text;
}

/* Smooth transitions */
* {
  transition: background-color 0.2s ease, border-color 0.2s ease, color 0.2s ease;
}

/* Remove transition from animations */
[data-framer-motion] {
  transition: none !important;
}

/* Input focus styles */
input:focus,
textarea:focus {
  outline: none;
  box-shadow: 0 0 0 3px rgba(107, 93, 211, 0.1);
}
```

```
/* Selection color */
::selection {
  background-color: rgba(107, 93, 211, 0.2);
  color: var(--text-primary);
}
```

🎨 TAILWIND CONFIG

File: `tailwind.config.ts`

typescript

```
import type { Config } from 'tailwindcss'

const config: Config = {
  content: [
    './pages/**/*.{js,ts,jsx,tsx,mdx}',
    './components/**/*.{js,ts,jsx,tsx,mdx}',
    './app/**/*.{js,ts,jsx,tsx,mdx}',
  ],
  theme: {
    extend: {
      colors: {
        routix: {
          purple: '#6B5DD3',
          'purple-light': '#8B7AFF',
          blue: '#7AA3FF',
        },
        text: {
          primary: '#2D2A4A',
          secondary: '#6B6B8D',
          muted: '#A0A0B8',
        },
      },
      backgroundImage: {
        'gradient-main': 'linear-gradient(135deg, #E0C3FC 0%, #D5E1FF 25%, #E8F4FF 50%, #FFE8F5 75%, #FFE5E5 100%)',
        'gradient-purple': 'linear-gradient(135deg, #6B5DD3 0%, #8B7AFF 100%)',
        'gradient-blue': 'linear-gradient(135deg, #7AA3FF 0%, #A8C5FF 100%)',
      },
      borderRadius: {
        'xl': '1rem',
        '2xl': '1.25rem',
        '3xl': '1.5rem',
        '4xl': '2rem',
      },
      backdropBlur: {
        'xl': '20px',
      },
      boxShadow: {
        'glass': '0 8px 32px rgba(31, 38, 135, 0.08)',
        'glass-hover': '0 12px 40px rgba(31, 38, 135, 0.12)',
        'button': '0 4px 15px rgba(107, 93, 211, 0.3)',
        'button-hover': '0 6px 20px rgba(107, 93, 211, 0.4)',
      },
      animation: {
```

```
        'float': 'float 6s ease-in-out infinite',
        'pulse-slow': 'pulse 3s cubic-bezier(0.4, 0, 0.6, 1) infinite',
      },
      keyframes: {
        float: {
          '0%, 100%': { transform: 'translateY(0px)' },
          '50%': { transform: 'translateY(-20px)' },
        },
      },
    },
  },
  plugins: [],
}

export default config
```

🔌 API CLIENT

File: `lib/api.ts`

typescript

```typescript
import axios from 'axios'

const api = axios.create({
  baseURL: process.env.NEXT_PUBLIC_API_URL || 'http://localhost:8000',
  headers: {
    'Content-Type': 'application/json',
  },
})

// Request interceptor for auth token
api.interceptors.request.use(
  (config) => {
    const token = localStorage.getItem('token')
    if (token) {
      config.headers.Authorization = `Bearer ${token}`
    }
    return config
  },
  (error) => Promise.reject(error)
)

// Response interceptor for error handling
api.interceptors.response.use(
  (response) => response,
  (error) => {
    if (error.response?.status === 401) {
      // Redirect to login
      localStorage.removeItem('token')
      window.location.href = '/login'
    }
    return Promise.reject(error)
  }
)

// API Methods
export const apiClient = {
  // Auth
  login: (email: string, password: string) =>
    api.post('/api/v1/auth/login', { email, password }),

  register: (data: any) =>
    api.post('/api/v1/auth/register', data),
```

```
// Generation
generate: (data: {
  prompt: string
  algorithm_id?: string
  user_face_url?: string
  user_logo_url?: string
  custom_text?: string
}) =>
  api.post('/api/v1/generate', data),

getGenerationStatus: (id: string) =>
  api.get(`/api/v1/generate/${id}/status`),

getGenerationResult: (id: string) =>
  api.get(`/api/v1/generate/${id}/result`),

getHistory: (page = 1, per_page = 20) =>
  api.get('/api/v1/generate/history', { params: { page, per_page } }),

// Algorithms
getAlgorithms: () =>
  api.get('/api/v1/algorithms'),

// Assets
uploadAsset: (file: File, type: 'face' | 'logo' | 'product') => {
  const formData = new FormData()
  formData.append('file', file)
  formData.append('asset_type', type)
  return api.post('/api/v1/user/assets/upload', formData, {
    headers: { 'Content-Type': 'multipart/form-data' }
  })
},

getUserAssets: () =>
  api.get('/api/v1/user/assets'),

// Admin - Templates
uploadTemplates: (files: File[], metadata?: any) => {
  const formData = new FormData()
  files.forEach(file => formData.append('files', file))
  if (metadata) {
    formData.append('metadata', JSON.stringify(metadata))
  }
  return api.post('/api/v1/admin/templates/bulk-upload', formData, {
```

```
    headers: { 'Content-Type': 'multipart/form-data' }
  })
},

getTemplates: (params?: any) =>
  api.get('/api/v1/admin/templates', { params }),

analyzeTemplate: (id: string) =>
  api.post(`/api/v1/admin/templates/${id}/analyze`),

batchAnalyze: (template_ids: string[]) =>
  api.post('/api/v1/admin/templates/batch-analyze', { template_ids }),

deleteTemplate: (id: string) =>
  api.delete(`/api/v1/admin/templates/${id}`),

// Admin - Dashboard
getAdminStats: () =>
  api.get('/api/v1/admin/dashboard/stats'),

getAdminCharts: (period = '7d') =>
  api.get('/api/v1/admin/dashboard/charts', { params: { period } }),

getRealtimeActivity: () =>
  api.get('/api/v1/admin/dashboard/realtime'),

// Admin - Users
getUsers: (params?: any) =>
  api.get('/api/v1/admin/users', { params }),

getUserDetails: (id: string) =>
  api.get(`/api/v1/admin/users/${id}`),

updateUserCredits: (id: string, amount: number) =>
  api.patch(`/api/v1/admin/users/${id}/credits`, { amount }),

banUser: (id: string, reason: string) =>
  api.patch(`/api/v1/admin/users/${id}/ban`, { reason }),

// Admin - Algorithms
createAlgorithm: (data: any) =>
  api.post('/api/v1/admin/algorithms', data),

updateAlgorithm: (id: string, data: any) =>
```

```
    api.put(`/api/v1/admin/algorithms/${id}`, data),

  deleteAlgorithm: (id: string) =>
    api.delete(`/api/v1/admin/algorithms/${id}`),
}

export default api
```

🪝 CUSTOM HOOKS

File: `hooks/usePolling.ts`

typescript

```
import { useEffect, useState } from 'react'
import { useQuery } from '@tanstack/react-query'
import { apiClient } from '@/lib/api'

export function useGenerationPolling(generationId: string) {
  const [isCompleted, setIsCompleted] = useState(false)

  const { data, isLoading, error } = useQuery({
    queryKey: ['generation-status', generationId],
    queryFn: () => apiClient.getGenerationStatus(generationId),
    refetchInterval: (data) => {
      // Stop polling when completed or failed
      if (data?.data?.status === 'completed' || data?.data?.status === 'failed') {
        setIsCompleted(true)
        return false
      }
      return 2000 // Poll every 2 seconds
    },
    enabled: !!generationId && !isCompleted,
  })

  return {
    data: data?.data,
    isLoading,
    error,
    isCompleted,
  }
}
```

File: `hooks/useGeneration.ts`

typescript

```typescript
import { useMutation, useQueryClient } from '@tanstack/react-query'
import { apiClient } from '@/lib/api'
import { toast } from 'sonner'

export function useGeneration() {
  const queryClient = useQueryClient()

  const mutation = useMutation({
    mutationFn: apiClient.generate,
    onSuccess: (data) => {
      toast.success('Generation started!')
      queryClient.invalidateQueries({ queryKey: ['generations'] })
    },
    onError: (error: any) => {
      toast.error(error.response?.data?.message || 'Generation failed')
    },
  })

  return mutation
}
```

📦 PACKAGE.JSON

json

```
{
  "name": "routix-web",
  "version": "1.0.0",
  "private": true,
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint"
  },
  "dependencies": {
    "next": "14.2.5",
    "react": "18.3.1",
    "react-dom": "18.3.1",
    "typescript": "5.3.3",
    "@tanstack/react-query": "^5.51.1",
    "axios": "^1.7.2",
    "zustand": "^4.5.4",
    "framer-motion": "^11.3.19",
    "react-hook-form": "^7.52.1",
    "zod": "^3.23.8",
    "lucide-react": "^0.414.0",
    "sonner": "^1.5.0",
    "react-dropzone": "^14.2.3",
    "recharts": "^2.12.7",
    "clsx": "^2.1.1",
    "tailwind-merge": "^2.4.0"
  },
  "devDependencies": {
    "@types/node": "20.14.12",
    "@types/react": "18.3.3",
    "@types/react-dom": "18.3.0",
    "autoprefixer": "^10.4.19",
    "postcss": "^8.4.39",
    "tailwindcss": "^3.4.6",
    "eslint": "8.57.0",
    "eslint-config-next": "14.2.5"
  }
}
```

🚀 ENVIRONMENT VARIABLES

File: `.env.local`

```bash
# API
NEXT_PUBLIC_API_URL=http://localhost:8000

# Auth (if using NextAuth)
NEXTAUTH_URL=http://localhost:3000
NEXTAUTH_SECRET=your-secret-key-here

# Optional: Analytics, etc.
NEXT_PUBLIC_GA_ID=
```

📋 IMPLEMENTATION CHECKLIST

Phase 1: Setup & Base Components ✅

```bash
1. Create Next.js project
2. Install dependencies
3. Setup Tailwind with custom config
4. Create base components:
   - GlassCard
   - GradientButton
   - Logo
   - Input
   - StatusBadge
5. Setup global styles
6. Test gradient background
```

Phase 2: Generation Interface (PRIORITY) ⭐

```bash
```

1. Build ChatInterface component
2. Build ProgressStatus component
3. Build ResultCard component
4. Implement generation flow
5. Add polling hook
6. Test complete flow
7. Add animations

Phase 3: Dashboard & History

bash

1. User dashboard layout
2. Sidebar navigation
3. Stats cards
4. History page with grid
5. Thumbnail cards
6. Filter & search

Phase 4: Admin Panel

bash

1. Admin dashboard
2. Template upload (drag & drop) ⭐
3. Template grid/list view
4. Bulk operations
5. User management
6. Algorithm management
7. Analytics charts

Phase 5: Authentication

bash

1. Login page
2. Register page
3. Auth context
4. Protected routes
5. Token management

Phase 6: Polish & Deploy

```
bash

1. Loading states
2. Error handling
3. Toast notifications
4. Responsive design
5. Performance optimization
6. Build & deploy
```

🎯 CRITICAL SUCCESS FACTORS

1. Visual Fidelity (MOST IMPORTANT)

- Gradient background MUST match reference images exactly
- Glassmorphism effect MUST be pixel-perfect
- Colors MUST use exact values from design system
- Animations MUST be smooth (60fps)

2. Generation Flow (CORE FEATURE)

- Chat-like interface MUST work flawlessly
- Real-time status updates MUST be smooth
- Progress messages MUST match reference ("Syncing with Routix Intelligence...")
- Completion animation MUST be delightful

3. Admin Panel (BUSINESS CRITICAL)

- Drag & drop MUST support multiple files
- Bulk upload MUST show progress
- Template grid MUST be fast with many items
- Analyze button MUST trigger backend correctly

4. Performance

- First paint < 1 second
- Smooth 60fps animations
- Optimized images

- Code splitting for admin panel

📞 API INTEGRATION GUIDE

All API calls go through the `apiClient` in `lib/api.ts`.

Example: Generate Thumbnail

```typescript
const { mutate, isLoading } = useGeneration()

mutate({
  prompt: "Gaming thumbnail with epic explosion",
  algorithm_id: "uuid-of-routix-v1",
  user_face_url: "https://...",
  custom_text: "EPIC GAME"
})
```

Example: Poll Status

```typescript
const { data, isCompleted } = useGenerationPolling(generationId)

// data.status will be: 'analyzing', 'generating', 'completed', 'failed'
// data.progress will be: 0-100
```

Example: Upload Templates (Admin)

```typescript
const uploadTemplates = async (files: File[]) => {
  const response = await apiClient.uploadTemplates(files, {
    category: ['tutorial'],
    tags: ['gaming', 'energetic']
  })

  // Then trigger analysis
  await apiClient.batchAnalyze(response.data.template_ids)
}
```

🎨 DESIGN TOKENS REFERENCE

typescript

```typescript
// Use these in your components
const tokens = {
  colors: {
    purple: '#6B5DD3',
    purpleLight: '#8B7AFF',
    blue: '#7AA3FF',
    textPrimary: '#2D2A4A',
    textSecondary: '#6B6B8D',
    textMuted: '#A0A0B8',
  },

  spacing: {
    xs: '8px',
    sm: '16px',
    md: '24px',
    lg: '32px',
    xl: '48px',
  },

  borderRadius: {
    sm: '12px',
    md: '16px',
    lg: '20px',
    xl: '24px',
  },

  shadows: {
    glass: '0 8px 32px rgba(31, 38, 135, 0.08)',
    button: '0 4px 15px rgba(107, 93, 211, 0.3)',
  }
}
```

✅ ACCEPTANCE CRITERIA

Before considering the project complete:

- [ ] Landing page loads with perfect gradient
- [ ] Login/Register works with backend

- [ ] Generation flow matches reference images EXACTLY
- [ ] Progress status updates in real-time
- [ ] Result displays correctly
- [ ] History page shows all generations
- [ ] Admin can drag & drop multiple templates
- [ ] Bulk upload shows progress
- [ ] Templates can be analyzed
- [ ] Algorithm management works
- [ ] User management functions correctly
- [ ] All animations are smooth
- [ ] Mobile responsive (bonus)
- [ ] No console errors
- [ ] Fast loading times

🚨 COMMON PITFALLS TO AVOID

1. **DON'T use hardcoded colors** - Use design tokens
2. **DON'T skip backdrop-filter** - Essential for glass effect
3. **DON'T forget -webkit-backdrop-filter** - Safari support
4. **DON'T use heavy images** - Optimize everything
5. **DON'T skip loading states** - Always show feedback
6. **DON'T forget error boundaries** - Handle all errors gracefully
7. **DON'T skip polling cleanup** - Memory leaks are real
8. **DON'T ignore TypeScript errors** - Fix them all

🎓 LEARNING RESOURCES

If stuck on specific features:

- **Glassmorphism**: https://glassmorphism.com
- **Framer Motion**: https://www.framer.com/motion/
- **TanStack Query**: https://tanstack.com/query/latest
- **Next.js App Router**: https://nextjs.org/docs
- **Tailwind CSS**: https://tailwindcss.com/docs

📝 FINAL NOTES

This is a COMPLETE specification. Everything you need to build the Routix frontend is here:

✅ Exact design system from reference images ✅ Complete component library ✅ Full API integration guide
✅ Step-by-step implementation plan ✅ All necessary dependencies ✅ Performance best practices

Start with Phase 2 (Generation Interface) - it's the most important feature and will set the tone for the entire application.

Good luck! 🚀