PRD Lovable 02: Design System e Componentes - LoL Engine

🎯 Objetivo para Lovable AI

Criar um design system completo e profissional com componentes reutilizáveis, tema premium e identidade visual consistente para o LoL Engine.

📋 Checklist de Implementação

✅ 1. Design Tokens

- [ ] Configurar cores do tema
- [ ] Definir tipografia
- [ ] Configurar espaçamentos
- [ ] Definir sombras e bordas

✅ 2. Componentes Base

- [ ] Button com variantes
- [ ] Input e Form components
- [ ] Card e Container
- [ ] Badge e Status indicators

✅ 3. Componentes Avançados

- [ ] Modal e Dialog
- [ ] Dropdown e Select
- [ ] Progress Bar
- [ ] Loading Spinner

✅ 4. Layout Components

- [ ] Header e Navigation
- [ ] Sidebar
- [ ] Footer

- [ ] Grid System

🎨 Design Tokens (TailwindCSS Config)

JavaScript

```javascript
// tailwind.config.js
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: [
    './src/pages/**/*.{js,ts,jsx,tsx,mdx}',
    './src/components/**/*.{js,ts,jsx,tsx,mdx}',
    './src/app/**/*.{js,ts,jsx,tsx,mdx}',
  ],
  theme: {
    extend: {
      colors: {
        // Primary Colors
        primary: {
          navy: '#2D3748',
          white: '#FFFFFF',
        },
        // Secondary Colors
        secondary: {
          lightGray: '#F7FAFC',
          mediumGray: '#E2E8F0',
        },
        // Accent Colors
        accent: {
          mintGreen: '#9AE6B4',
          softBlue: '#BEE3F8',
        },
        // Text Colors
        text: {
          darkGray: '#2D3748',
          mediumGray: '#718096',
          lightGray: '#A0AEC0',
        },
        // Status Colors
        status: {
          success: '#48BB78',
          warning: '#ED8936',
          error: '#F56565',
          info: '#4299E1',
        },
        // Evidence Colors
        evidence: {
```

```
          strong: '#48BB78',
          moderate: '#ED8936',
          limited: '#ECC94B',
          inconclusive: '#A0AEC0',
        }
      },
      fontFamily: {
        sans: ['Inter', 'system-ui', 'sans-serif'],
        heading: ['Poppins', 'system-ui', 'sans-serif'],
      },
      fontSize: {
        'xs': '0.75rem',
        'sm': '0.875rem',
        'base': '1rem',
        'lg': '1.125rem',
        'xl': '1.25rem',
        '2xl': '1.5rem',
        '3xl': '1.875rem',
        '4xl': '2.25rem',
      },
      spacing: {
        '18': '4.5rem',
        '88': '22rem',
      },
      boxShadow: {
        'soft': '0 2px 15px -3px rgba(0, 0, 0, 0.07), 0 10px 20px -2px
rgba(0, 0, 0, 0.04)',
        'medium': '0 4px 25px -5px rgba(0, 0, 0, 0.1), 0 10px 10px -5px
rgba(0, 0, 0, 0.04)',
        'strong': '0 10px 40px -10px rgba(0, 0, 0, 0.15), 0 2px 10px -2px
rgba(0, 0, 0, 0.05)',
      },
      borderRadius: {
        'xl': '1rem',
        '2xl': '1.5rem',
      }
    },
  },
  plugins: [],
}
```

🧩 Componentes Base

Button Component

TypeScript

```tsx
// components/ui/Button.tsx
import { ButtonHTMLAttributes, forwardRef } from "react"
import { cva, type VariantProps } from "class-variance-authority"
import { cn } from "@/lib/utils"

const buttonVariants = cva(
  "inline-flex items-center justify-center rounded-lg text-sm font-medium
transition-all duration-200 focus-visible:outline-none focus-visible:ring-2
focus-visible:ring-offset-2 disabled:opacity-50 disabled:pointer-events-
none",
  {
    variants: {
      variant: {
        primary: "bg-primary-navy text-primary-white hover:bg-primary-
navy/90 shadow-md hover:shadow-lg",
        secondary: "bg-secondary-lightGray text-text-darkGray hover:bg-
secondary-mediumGray",
        accent: "bg-accent-mintGreen text-primary-navy hover:bg-accent-
mintGreen/90 shadow-md",
        outline: "border-2 border-primary-navy text-primary-navy hover:bg-
primary-navy hover:text-primary-white",
        ghost: "text-text-darkGray hover:bg-secondary-lightGray",
        success: "bg-status-success text-primary-white hover:bg-status-
success/90",
        warning: "bg-status-warning text-primary-white hover:bg-status-
warning/90",
        error: "bg-status-error text-primary-white hover:bg-status-error/90",
      },
      size: {
        sm: "h-8 px-3 text-xs",
        md: "h-10 px-4 text-sm",
        lg: "h-12 px-6 text-base",
        xl: "h-14 px-8 text-lg",
      }
    },
    defaultVariants: {
      variant: "primary",
      size: "md"
    }
  }
)

export interface ButtonProps
  extends ButtonHTMLAttributes<HTMLButtonElement>,
    VariantProps<typeof buttonVariants> {}

const Button = forwardRef<HTMLButtonElement, ButtonProps>(
```

```
  ({ className, variant, size, ...props }, ref) => {
    return (
      <button
        className={cn(buttonVariants({ variant, size, className }))}
        ref={ref}
        {...props}
      />
    )
  }
)
Button.displayName = "Button"

export { Button, buttonVariants }
```

Input Component

TypeScript

```
// components/ui/Input.tsx
import { InputHTMLAttributes, forwardRef } from "react"
import { cva, type VariantProps } from "class-variance-authority"
import { cn } from "@/lib/utils"

const inputVariants = cva(
  "flex w-full rounded-lg border bg-primary-white px-3 py-2 text-sm
transition-colors file:border-0 file:bg-transparent file:text-sm file:font-
medium placeholder:text-text-lightGray focus-visible:outline-none focus-
visible:ring-2 focus-visible:ring-offset-2 disabled:cursor-not-allowed
disabled:opacity-50",
  {
    variants: {
      variant: {
        default: "border-secondary-mediumGray focus-visible:ring-accent-
mintGreen",
        error: "border-status-error focus-visible:ring-status-error",
        success: "border-status-success focus-visible:ring-status-success",
      },
      size: {
        sm: "h-8 px-2 text-xs",
        md: "h-10 px-3 text-sm",
        lg: "h-12 px-4 text-base",
      }
    },
    defaultVariants: {
      variant: "default",
      size: "md"
    }
```

```
  }
)

export interface InputProps
  extends InputHTMLAttributes<HTMLInputElement>,
    VariantProps<typeof inputVariants> {}

const Input = forwardRef<HTMLInputElement, InputProps>(
  ({ className, variant, size, type, ...props }, ref) => {
    return (
      <input
        type={type}
        className={cn(inputVariants({ variant, size, className }))}
        ref={ref}
        {...props}
      />
    )
  }
)
Input.displayName = "Input"

export { Input, inputVariants }
```

Card Component

TypeScript

```
// components/ui/Card.tsx
import { HTMLAttributes, forwardRef } from "react"
import { cva, type VariantProps } from "class-variance-authority"
import { cn } from "@/lib/utils"

const cardVariants = cva(
  "rounded-xl border bg-primary-white text-text-darkGray",
  {
    variants: {
      variant: {
        default: "border-secondary-mediumGray",
        elevated: "border-secondary-mediumGray shadow-soft",
        premium: "border-accent-mintGreen shadow-medium bg-gradient-to-br
from-primary-white to-secondary-lightGray",
        success: "border-status-success bg-status-success/5",
        warning: "border-status-warning bg-status-warning/5",
        error: "border-status-error bg-status-error/5",
      },
      padding: {
        none: "p-0",
```

```
        sm: "p-3",
        md: "p-4",
        lg: "p-6",
        xl: "p-8",
      }
    },
    defaultVariants: {
      variant: "default",
      padding: "md"
    }
  }
)

export interface CardProps
  extends HTMLAttributes<HTMLDivElement>,
    VariantProps<typeof cardVariants> {}

const Card = forwardRef<HTMLDivElement, CardProps>(
  ({ className, variant, padding, ...props }, ref) => {
    return (
      <div
        ref={ref}
        className={cn(cardVariants({ variant, padding, className }))}
        {...props}
      />
    )
  }
)
Card.displayName = "Card"

const CardHeader = forwardRef<HTMLDivElement, HTMLAttributes<HTMLDivElement>>
(
  ({ className, ...props }, ref) => (
    <div
      ref={ref}
      className={cn("flex flex-col space-y-1.5 pb-4", className)}
      {...props}
    />
  )
)
CardHeader.displayName = "CardHeader"

const CardTitle = forwardRef<HTMLParagraphElement,
HTMLAttributes<HTMLHeadingElement>>(
  ({ className, ...props }, ref) => (
    <h3
      ref={ref}
      className={cn("text-xl font-semibold leading-none tracking-tight font-
```

```
heading", className)}
      {...props}
    />
  )
)
CardTitle.displayName = "CardTitle"

const CardContent = forwardRef<HTMLDivElement,
HTMLAttributes<HTMLDivElement>>(
  ({ className, ...props }, ref) => (
    <div ref={ref} className={cn("", className)} {...props} />
  )
)
CardContent.displayName = "CardContent"

export { Card, CardHeader, CardTitle, CardContent, cardVariants }
```

Badge Component

TypeScript

```
// components/ui/Badge.tsx
import { HTMLAttributes, forwardRef } from "react"
import { cva, type VariantProps } from "class-variance-authority"
import { cn } from "@/lib/utils"

const badgeVariants = cva(
  "inline-flex items-center rounded-full border px-2.5 py-0.5 text-xs font-
semibold transition-colors focus:outline-none focus:ring-2 focus:ring-ring
focus:ring-offset-2",
  {
    variants: {
      variant: {
        default: "border-transparent bg-secondary-mediumGray text-text-
darkGray",
        secondary: "border-transparent bg-secondary-lightGray text-text-
mediumGray",
        success: "border-transparent bg-status-success text-primary-white",
        warning: "border-transparent bg-status-warning text-primary-white",
        error: "border-transparent bg-status-error text-primary-white",
        info: "border-transparent bg-status-info text-primary-white",
        accent: "border-transparent bg-accent-mintGreen text-primary-navy",
        outline: "border-secondary-mediumGray text-text-darkGray",
        // Evidence levels
        strong: "border-transparent bg-evidence-strong text-primary-white",
        moderate: "border-transparent bg-evidence-moderate text-primary-
white",
```

```
        limited: "border-transparent bg-evidence-limited text-primary-navy",
        inconclusive: "border-transparent bg-evidence-inconclusive text-
primary-white",
      },
      size: {
        sm: "px-2 py-0.5 text-xs",
        md: "px-2.5 py-0.5 text-xs",
        lg: "px-3 py-1 text-sm",
      }
    },
    defaultVariants: {
      variant: "default",
      size: "md"
    }
  }
)

export interface BadgeProps
  extends HTMLAttributes<HTMLDivElement>,
    VariantProps<typeof badgeVariants> {}

function Badge({ className, variant, size, ...props }: BadgeProps) {
  return (
    <div className={cn(badgeVariants({ variant, size }), className)}
{...props} />
  )
}

// Evidence Badge específico
interface EvidenceBadgeProps {
  level: 'strong' | 'moderate' | 'limited' | 'inconclusive'
  size?: 'sm' | 'md' | 'lg'
  className?: string
}

function EvidenceBadge({ level, size = 'md', className }:
EvidenceBadgeProps) {
  const labels = {
    strong: 'Evidência Forte',
    moderate: 'Evidência Moderada',
    limited: 'Evidência Limitada',
    inconclusive: 'Evidência Inconclusiva'
  }

  return (
    <Badge variant={level} size={size} className={className}>
      {labels[level]}
    </Badge>
```

```
  )
}

export { Badge, EvidenceBadge, badgeVariants }
```

Progress Bar Component

TypeScript

```typescript
// components/ui/ProgressBar.tsx
import { HTMLAttributes, forwardRef } from "react"
import { cva, type VariantProps } from "class-variance-authority"
import { cn } from "@/lib/utils"

const progressVariants = cva(
  "relative overflow-hidden rounded-full bg-secondary-lightGray",
  {
    variants: {
      size: {
        sm: "h-2",
        md: "h-3",
        lg: "h-4",
      }
    },
    defaultVariants: {
      size: "md"
    }
  }
)

const progressBarVariants = cva(
  "h-full w-full flex-1 bg-primary transition-all duration-300 ease-in-out",
  {
    variants: {
      variant: {
        default: "bg-accent-mintGreen",
        success: "bg-status-success",
        warning: "bg-status-warning",
        error: "bg-status-error",
        info: "bg-status-info",
      }
    },
    defaultVariants: {
      variant: "default"
    }
  }
)
```

```tsx
export interface ProgressBarProps
  extends HTMLAttributes<HTMLDivElement>,
    VariantProps<typeof progressVariants>,
    VariantProps<typeof progressBarVariants> {
  value: number
  max?: number
  showLabel?: boolean
  label?: string
}

const ProgressBar = forwardRef<HTMLDivElement, ProgressBarProps>(
  ({ className, size, variant, value, max = 100, showLabel, label, ...props
}, ref) => {
    const percentage = Math.min(Math.max((value / max) * 100, 0), 100)

    return (
      <div className="space-y-2">
        {showLabel && (
          <div className="flex justify-between text-sm">
            <span className="text-text-mediumGray">{label}</span>
            <span className="font-medium text-text-darkGray">
{Math.round(percentage)}%</span>
          </div>
        )}
        <div
          ref={ref}
          className={cn(progressVariants({ size, className }))}
          {...props}
        >
          <div
            className={cn(progressBarVariants({ variant }))}
            style={{ width: `${percentage}%` }}
          />
        </div>
      </div>
    )
  }
)
ProgressBar.displayName = "ProgressBar"

export { ProgressBar, progressVariants, progressBarVariants }
```

Loading Spinner Component

TypeScript

```tsx
// components/ui/LoadingSpinner.tsx
import { HTMLAttributes } from "react"
import { cva, type VariantProps } from "class-variance-authority"
import { cn } from "@/lib/utils"

const spinnerVariants = cva(
  "animate-spin rounded-full border-2 border-current border-t-transparent",
  {
    variants: {
      size: {
        sm: "h-4 w-4",
        md: "h-6 w-6",
        lg: "h-8 w-8",
        xl: "h-12 w-12",
      },
      color: {
        default: "text-accent-mintGreen",
        white: "text-primary-white",
        navy: "text-primary-navy",
        gray: "text-text-mediumGray",
      }
    },
    defaultVariants: {
      size: "md",
      color: "default"
    }
  }
)

export interface LoadingSpinnerProps
  extends HTMLAttributes<HTMLDivElement>,
    VariantProps<typeof spinnerVariants> {}

function LoadingSpinner({ className, size, color, ...props }:
LoadingSpinnerProps) {
  return (
    <div
      className={cn(spinnerVariants({ size, color }), className)}
      {...props}
    />
  )
}

// Loading com texto
interface LoadingWithTextProps {
  text?: string
  size?: 'sm' | 'md' | 'lg' | 'xl'
```

```
  color?: 'default' | 'white' | 'navy' | 'gray'
}

function LoadingWithText({ text = "Carregando...", size = "md", color =
"default" }: LoadingWithTextProps) {
  return (
    <div className="flex items-center space-x-3">
      <LoadingSpinner size={size} color={color} />
      <span className="text-sm text-text-mediumGray">{text}</span>
    </div>
  )
}

export { LoadingSpinner, LoadingWithText, spinnerVariants }
```

🏗️ Layout Components

Header Component

TypeScript

```
// components/layout/Header.tsx
"use client"

import { useState } from "react"
import { useSession, signOut } from "next-auth/react"
import { Button } from "@/components/ui/Button"
import { Badge } from "@/components/ui/Badge"

export function Header() {
  const { data: session } = useSession()
  const [isMenuOpen, setIsMenuOpen] = useState(false)

  return (
    <header className="sticky top-0 z-50 w-full border-b border-secondary-
mediumGray bg-primary-white/95 backdrop-blur supports-[backdrop-filter]:bg-
primary-white/60">
      <div className="container mx-auto px-4">
        <div className="flex h-16 items-center justify-between">
          {/* Logo */}
          <div className="flex items-center space-x-2">
            <div className="h-8 w-8 rounded-lg bg-gradient-to-br from-accent-
mintGreen to-accent-softBlue flex items-center justify-center">
              <span className="text-primary-navy font-bold text-lg">L</span>
            </div>
            <span className="text-xl font-bold font-heading text-primary-
```

```
navy">
              LoL Engine
            </span>
            <Badge variant="accent" size="sm">Beta</Badge>
          </div>

          {/* Navigation */}
          <nav className="hidden md:flex items-center space-x-6">
            <a href="/dashboard" className="text-text-darkGray hover:text-
accent-mintGreen transition-colors">
              Dashboard
            </a>
            <a href="/analysis" className="text-text-darkGray hover:text-
accent-mintGreen transition-colors">
              Análise
            </a>
            <a href="/progress" className="text-text-darkGray hover:text-
accent-mintGreen transition-colors">
              Progresso
            </a>
            <a href="/supplements" className="text-text-darkGray hover:text-
accent-mintGreen transition-colors">
              Suplementos
            </a>
          </nav>

          {/* User Menu */}
          <div className="flex items-center space-x-4">
            {session ? (
              <div className="flex items-center space-x-3">
                <div className="hidden md:block text-right">
                  <div className="text-sm font-medium text-text-darkGray">
                    {session.user?.name}
                  </div>
                  <div className="text-xs text-text-mediumGray">
                    {session.user?.email}
                  </div>
                </div>
                <Button
                  variant="outline"
                  size="sm"
                  onClick={() => signOut()}
                >
                  Sair
                </Button>
              </div>
            ) : (
              <div className="flex items-center space-x-2">
```

```
                <Button variant="ghost" size="sm">
                  <a href="/login">Entrar</a>
                </Button>
                <Button variant="primary" size="sm">
                  <a href="/register">Criar Conta</a>
                </Button>
              </div>
            )}
          </div>
        </div>
      </div>
    </header>
  )
}
```

Footer Component

TypeScript

```
// components/layout/Footer.tsx
export function Footer() {
  return (
    <footer className="border-t border-secondary-mediumGray bg-secondary-
lightGray">
      <div className="container mx-auto px-4 py-8">
        <div className="grid grid-cols-1 md:grid-cols-4 gap-8">
          {/* Logo e descrição */}
          <div className="space-y-4">
            <div className="flex items-center space-x-2">
              <div className="h-6 w-6 rounded bg-gradient-to-br from-accent-
mintGreen to-accent-softBlue flex items-center justify-center">
                <span className="text-primary-navy font-bold text-
sm">L</span>
              </div>
              <span className="text-lg font-bold font-heading text-primary-
navy">
                LoL Engine
              </span>
            </div>
            <p className="text-sm text-text-mediumGray">
              Recomendações de suplementos personalizadas com IA para
otimizar sua saúde e performance.
            </p>
          </div>

          {/* Links */}
          <div>
```

```
            <h4 className="font-semibold text-text-darkGray mb-
4">Produto</h4>
            <ul className="space-y-2 text-sm text-text-mediumGray">
              <li><a href="/features" className="hover:text-accent-mintGreen
transition-colors">Funcionalidades</a></li>
              <li><a href="/pricing" className="hover:text-accent-mintGreen
transition-colors">Preços</a></li>
              <li><a href="/science" className="hover:text-accent-mintGreen
transition-colors">Ciência</a></li>
            </ul>
          </div>

          <div>
            <h4 className="font-semibold text-text-darkGray mb-
4">Suporte</h4>
            <ul className="space-y-2 text-sm text-text-mediumGray">
              <li><a href="/help" className="hover:text-accent-mintGreen
transition-colors">Central de Ajuda</a></li>
              <li><a href="/contact" className="hover:text-accent-mintGreen
transition-colors">Contato</a></li>
              <li><a href="/faq" className="hover:text-accent-mintGreen
transition-colors">FAQ</a></li>
            </ul>
          </div>

          <div>
            <h4 className="font-semibold text-text-darkGray mb-4">Legal</h4>
            <ul className="space-y-2 text-sm text-text-mediumGray">
              <li><a href="/privacy" className="hover:text-accent-mintGreen
transition-colors">Privacidade</a></li>
              <li><a href="/terms" className="hover:text-accent-mintGreen
transition-colors">Termos</a></li>
              <li><a href="/disclaimer" className="hover:text-accent-
mintGreen transition-colors">Disclaimer</a></li>
            </ul>
          </div>
        </div>

        <div className="border-t border-secondary-mediumGray mt-8 pt-8 text-
center text-sm text-text-mediumGray">
          <p>&copy; 2025 LoL Engine. Todos os direitos reservados.</p>
        </div>
      </div>
    </footer>
  )
}
```

🎯 Instruções Específicas para Lovable

1. **Configure o TailwindCSS com os tokens exatos mostrados**
2. **Implemente todos os componentes na ordem: Button → Input → Card → Badge → ProgressBar → LoadingSpinner**
3. **Teste cada componente individualmente**
4. **Crie uma página de showcase para visualizar todos os componentes**
5. **Implemente Header e Footer por último**

✅ Critérios de Sucesso

- [ ] Todos os componentes renderizam corretamente
- [ ] Design tokens aplicados consistentemente
- [ ] Componentes responsivos em mobile
- [ ] Variantes funcionando (cores, tamanhos)
- [ ] Header e Footer integrados

📝 Notas para Lovable

- Use exatamente estas cores e espaçamentos
- Implemente todas as variantes mostradas
- Teste responsividade em diferentes tamanhos
- Mantenha consistência visual em todos os componentes