Implement Neumorphic Design System with HeroUI

Objective

Transform our existing HeroUI-based app into a cohesive neumorphic design system. Focus on creating depth and tactile feel through shadows and subtle gradients while maintaining all existing functionality.

Design Philosophy

- **Sunken elements**: Inputs, inactive states, content wells (look cut into the surface)
- **Raised elements**: Buttons, active states, important UI elements (appear elevated)
- **Flat elements**: Background surfaces, subtle secondary elements
- **Highly elevated**: Modals, dropdowns, overlays (strong depth hierarchy)

Implementation Requirements

Phase 1: Core Shadow System

Update `tailwind.config.js` to add neumorphic shadow utilities for dark mode:

```javascript
boxShadow: {
  // Main neumorphic shadows for dark backgrounds
  'neu-raised': '4px 4px 8px rgba(0, 0, 0, 0.3), -4px -4px 8px rgba(255, 255, 255, 0.05
  'neu-raised-lg': '8px 8px 16px rgba(0, 0, 0, 0.3), -8px -8px 16px rgba(255, 255, 255
  'neu-inset': 'inset 4px 4px 8px rgba(0, 0, 0, 0.3), inset -4px -4px 8px rgba(255, 255
  'neu-inset-lg': 'inset 8px 8px 16px rgba(0, 0, 0, 0.3), inset -8px -8px 16px rgba(255
  'neu-flat': '2px 2px 4px rgba(0, 0, 0, 0.2), -2px -2px 4px rgba(255, 255, 255, 0.03)

  // Interactive states
  'neu-raised-hover': '6px 6px 12px rgba(0, 0, 0, 0.4), -6px -6px 12px rgba(255, 255, 2
  'neu-pressed': 'inset 2px 2px 4px rgba(0, 0, 0, 0.4), inset -2px -2px 4px rgba(255, 2
}

backgroundImage: {
  'neu-surface': 'linear-gradient(145deg, rgba(255,255,255,0.02), rgba(0,0,0,0.1))',
  'neu-elevated': 'linear-gradient(145deg, rgba(255,255,255,0.03), rgba(0,0,0,0.05))',
  'neu-sunken': 'linear-gradient(145deg, rgba(0,0,0,0.1), rgba(255,255,255,0.02))',
}
```

Phase 2: Primary Component Transformations

1. Input Elements (Sunken/Inset Appearance)

Create neumorphic variants for all input components using `extendVariants`:

- **Input fields**: Sunken appearance with `shadow-neu-inset`
- **Focus state**: Deeper inset with `shadow-neu-inset-lg`
- **Background**: Use `bg-neu-sunken` gradient
- **Remove borders**: Set `border-0` for clean neumorphic look

2. Buttons (Raised Appearance)

Transform buttons to appear elevated:

- **Primary buttons**: Raised with `shadow-neu-raised`
- **Hover state**: More elevated with `shadow-neu-raised-hover` and slight `translate-y`
- **Active/pressed**: Sunken with `shadow-neu-pressed` (like physical button press)
- **Secondary buttons**: Subtle elevation with `shadow-neu-flat`

3. Cards (Flexible Elevation)

Create card variants for different use cases:

- **Elevated cards**: Default raised appearance for content containers
- **Sunken cards**: For form areas, content wells, input groups
- **Flat cards**: For subtle content separation
- **Hover animations**: Gentle elevation increase for interactive cards

Phase 3: Extended Component System

Apply neumorphic principles to additional components:

Interactive Elements

- **Switches**: Sunken track, raised thumb
- **Checkboxes**: Sunken when unchecked, raised when checked
- **Radio buttons**: Similar sunken/raised pattern
- **Progress bars**: Sunken track, slightly raised progress fill

Navigation & Layout

- **Tabs**: Sunken inactive, raised active
- **Modal overlays**: Highly elevated with strong shadows
- **Chips/badges**: Small raised elements
- **Dividers**: Subtle inset lines

Phase 4: Interaction States & Animations

Ensure all neumorphic elements have appropriate feedback:

- **Hover**: Smooth shadow transitions, subtle elevation changes
- **Focus**: Enhanced shadow depth for accessibility
- **Active/Pressed**: Quick transition to sunken state
- **Disabled**: Flattened appearance with reduced contrast
- **Loading**: Subtle pulsing of shadows

Technical Implementation Notes

HeroUI Integration

- Use `extendVariants` to create neumorphic versions of HeroUI components
- Leverage HeroUI's slot-based styling with `classNames` prop
- Maintain all existing accessibility features and props
- Preserve component APIs - only modify visual appearance

Performance Considerations

- Use CSS transitions for smooth shadow animations
- Keep shadow complexity reasonable for performance
- Ensure animations don't impact PWA performance

Consistency Guidelines

- Maintain consistent shadow directions (light source from top-left)
- Use established depth hierarchy throughout the app
- Keep shadow blur values proportional
- Ensure sufficient contrast for accessibility

Expected Outcome

A cohesive neumorphic design system that:

- Makes inputs feel naturally recessed into the interface
- Makes buttons feel tactile and pressable
- Creates clear visual hierarchy through elevation
- Maintains all existing functionality and accessibility
- Provides smooth, satisfying micro-interactions
- Feels modern and premium while being highly usable

Testing Requirements

- Verify all interactive elements maintain proper focus states
- Test shadow rendering performance across devices
- Ensure accessibility contrast requirements are met
- Validate that PWA functionality remains intact
- Check component responsiveness across screen sizes

Start with the shadow system and core components (Input, Button, Card), then systematically apply the neumorphic principles to the rest of the interface.