Neurodiversity SDK Development Prompt

Project Overview

Build a comprehensive, open-source SDK called "NeuroAdapt SDK" that enables developers to seamlessly integrate neurodiversity-focused accessibility features into AI, quantum computing, and VR/AR applications. The SDK should provide plug-and-play components, APIs, and tools that make it simple to create experiences that are accessible and comfortable for neurodivergent users, including those with autism, ADHD, dyslexia, sensory processing differences, and other neurological variations.

Vision Statement

Create the industry standard toolkit for neurodiversity inclusion in emerging technologies, ensuring that as we advance into AI, quantum, and immersive computing, no one is left behind due to neurological differences.

Technical Architecture

Core Stack

```
Language: TypeScript (with Python bindings for AI/ML components)
Build System: Vite + Rollup for tree-shakeable modules
Testing: Jest + Playwright for unit and integration tests
Documentation: TypeDoc + Docusaurus for interactive docs
Package Management: npm/yarn with monorepo structure (Lerna/Nx)
CI/CD: GitHub Actions with automated accessibility testing
```

Platform Support

- **Web**: ES modules, React/Vue/Angular adapters
- **VR/AR**: Unity SDK, Unreal Engine plugin, WebXR
- **AI Platforms**: TensorFlow.js, PyTorch, OpenAI API integration
- **Quantum**: Qiskit, Cirq, Q# integration layers
- **Mobile**: React Native, Flutter plugins
- **Desktop**: Electron support

SDK Architecture

1. Core Modules

A. Sensory Management Module

```
interface SensoryProfile {
  visualSensitivity: {
    brightness: number; // 0-1
    contrast: number; // 0-1
    colorTemp: number; // 2700K-6500K
    motionTolerance: 'none' | 'minimal' | 'moderate' | 'full';
    flickerSensitivity: boolean;
  };
  auditorySensitivity: {
    volumePreference: number; // 0-1
    frequencyFilters: FrequencyRange[];
    spatialAudioTolerance: boolean;
    suddenSoundSensitivity: boolean;
  };
  hapticSensitivity: {
    intensity: number; // 0-1
    patterns: HapticPattern[];
    enabled: boolean;
  };
}

// API Usage
const sensory = new SensoryManager();
await sensory.loadUserProfile();
sensory.applyToEnvironment(vrScene);
sensory.filterAudioStream(audioContext);
```

B. Cognitive Load Manager

```
interface CognitiveLoadSettings {
  informationDensity: 'minimal' | 'low' | 'medium' | 'high';
  pacing: {
    autoAdvance: boolean;
    minimumReadTime: number;
```

```
    breakReminders: boolean;
    breakInterval: number;
  };
  complexity: {
    abstractionLevel: number; // 1-5
    simultaneousTasks: number; // 1-3
    decisionPoints: number; // per interaction
  };
}

// Real-time cognitive load monitoring
const cognitive = new CognitiveLoadManager();
cognitive.on('overload', () => {
  // Automatically simplify interface
  // Reduce information density
  // Offer break
});
```

C. Communication Adapter

```
interface CommunicationPreferences {
  inputMethods: Array<'text' | 'voice' | 'gesture' | 'eyeTracking' |
'brainInterface'>;
  outputFormats: {
    textToSpeech: TTSConfig;
    visualCues: boolean;
    hapticFeedback: boolean;
    simplifiedLanguage: boolean;
  };
  processingTime: {
    responseDelay: number; // ms
    extendedInputTime: boolean;
    confirmationRequired: boolean;
  };
}

// Adaptive communication
const comm = new CommunicationAdapter();
comm.interpretInput(multiModalInput)
  .then(intent => comm.generateResponse(intent, userPrefs));
```

2. AI-Specific Features

A. Predictable AI Module

```
class PredictableAI {
  // Consistent interaction patterns
  async configureBehavior(config: AIBehaviorConfig): Promise<void>;

  // Explainable AI integration
  async explainDecision(
    decision: AIDecision,
    explanationLevel: 'simple' | 'detailed' | 'visual'
  ): Promise<Explanation>;

  // Controllable randomness
  setSeedForConsistency(seed: string): void;

  // Anxiety-reducing features
  previewNextAction(): ActionPreview;
  enableUndoBuffer(steps: number): void;
}

// Usage with OpenAI
const ai = new PredictableAI();
ai.wrapModel(openaiClient, {
  consistencyLevel: 'high',
  explanationDefault: 'simple',
  emotionalTone: 'supportive'
});
```

B. Attention Management for AI

```
interface AttentionGuidance {
  focusIndicators: {
    visual: VisualFocusConfig;
    audio: AudioCueConfig;
    haptic: HapticCueConfig;
  };
  distractionReduction: {
```

```
    hideNonEssential: boolean;
    progressiveDisclosure: boolean;
    focusMode: 'single' | 'guided' | 'free';
  };
}

// ADHD-friendly AI interactions
const attention = new AttentionManager();
attention.highlightImportant(aiResponse);
attention.breakDownComplexTask(task, userProfile);
```

3. VR/AR-Specific Features

A. Comfort Zone System

```
class VRComfortZone {
  // Personal space bubble
  setPersonalSpace(radius: number): void;
  enableProximityWarnings(config: ProximityConfig): void;

  // Safe retreat spaces
  createSafeSpace(triggers: SafeSpaceTrigger[]): SafeSpace;

  // Sensory regulation rooms
  generateCalmingEnvironment(preferences: SensoryProfile): VREnvironment;

  // Motion comfort
  teleportationMode: TeleportConfig;
  smoothLocomotion: LocomotionConfig;
  vestibularSupport: VestibularConfig;
}
```

B. VR Interface Adaptations

```
interface VRAccessibilityLayer {
  // Customizable UI positioning
  uiAnchoring: 'world' | 'head' | 'hand' | 'fixed';
  uiDistance: number; // meters from user
```

```
  // Interaction methods
  gazeSelection: GazeConfig;
  gestureRecognition: GestureConfig;
  voiceCommands: VoiceConfig;

  // Visual accommodations
  depthCueEnhancement: boolean;
  contrastBoost: number;
  colorBlindMode: ColorBlindType;

  // Subtitle system
  spatialSubtitles: SubtitleConfig;
  speakerIdentification: boolean;
}
```

4. Quantum Computing Accessibility

A. Quantum Concept Visualizer

```
class QuantumVisualizer {
  // Multi-sensory quantum state representation
  visualizeQubit(state: QuantumState): {
    visual: ThreeJSObject;
    audio: AudioRepresentation;
    haptic: HapticPattern;
  };

  // Step-by-step circuit building
  circuitBuilder: {
    guided: boolean;
    templates: CircuitTemplate[];
    validation: ValidationRule[];
  };

  // Simplified notation systems
  notationAdapter(
    notation: 'bra-ket' | 'matrix' | 'visual',
    complexity: 'basic' | 'intermediate' | 'advanced'
  ): NotationSystem;
}
```

B. Quantum Learning Aids

```
interface QuantumLearningSupport {
  // Concrete analogies for abstract concepts
  analogyEngine: AnalogyGenerator;

  // Interactive tutorials with pacing control
  tutorials: AdaptiveTutorial[];

  // Mistake-friendly environment
  sandboxMode: {
    infiniteUndo: boolean;
    mistakeExplanations: boolean;
    gentleCorrections: boolean;
  };
}
```

5. Universal Design Components

A. Predictable Interaction Patterns

```
class InteractionStandardizer {
  // Consistent gesture library
  static gestures = {
    select: GesturePattern,
    back: GesturePattern,
    menu: GesturePattern,
    help: GesturePattern
  };

  // Standardized feedback
  feedbackLibrary: {
    success: MultiModalFeedback,
    error: MultiModalFeedback,
    warning: MultiModalFeedback,
    info: MultiModalFeedback
  };

  // Predictable state changes
```

```
  transitionAnimations: {
    duration: number,
    easing: EasingFunction,
    previewMode: boolean
  };
}
```

B. Progressive Disclosure System

```
class ProgressiveDisclosure {
  // Information hierarchy
  setComplexityLevels(levels: ComplexityLevel[]): void;

  // Adaptive revealing
  revealBasedOn: {
    time: boolean;
    performance: boolean;
    preference: boolean;
    cognitiveLoad: boolean;
  };

  // Always-available simplification
  simplifyButton: SimplifyConfig;
  complexitySlider: SliderConfig;
}
```

6. Testing & Validation Tools

A. Neurodiversity Testing Suite

```
class NeuroTestSuite {
  // Automated accessibility checks
  async runAccessibilityTests(
    app: Application,
    profiles: NeurodiversityProfile[]
  ): Promise<TestReport>;

  // Cognitive load analysis
  analyzeCognitiveLoad(
```

```
    userFlow: UserFlow
  ): CognitiveLoadReport;

  // Sensory impact assessment
  assessSensoryImpact(
    scene: VRScene | ARScene
  ): SensoryImpactReport;

  // User comfort metrics
  measureComfortMetrics(
    session: SessionData
  ): ComfortMetrics;
}
```

B. Simulation Tools

```
class NeurodiversitySimulator {
  // Experience simulators
  simulateAutisticExperience(scene: Scene): SimulatedView;
  simulateADHDExperience(interface: UI): SimulatedView;
  simulateDyslexicExperience(text: TextContent): SimulatedView;

  // Sensory overload detector
  detectOverloadRisks(environment: Environment): Risk[];

  // Attention flow analyzer
  analyzeAttentionFlow(ui: UserInterface): AttentionMap;
}
```

7. Analytics & Insights

A. Usage Analytics

```
interface NeuroAnalytics {
  // Privacy-first analytics
  trackFeatureUsage(
    feature: Feature,
    anonymized: boolean
  ): void;
```

```
  // Comfort metrics
  comfortScore: {
    calculate(): number;
    factors: ComfortFactor[];
    improvements: Suggestion[];
  };

  // Adaptation effectiveness
  measureAdaptationSuccess(
    adaptation: Adaptation
  ): SuccessMetrics;
}
```

B. Personalization Engine

```
class PersonalizationEngine {
  // Learn from usage patterns
  async learnPreferences(
    interactions: Interaction[]
  ): Promise<PreferenceModel>;

  // Suggest optimizations
  suggestOptimizations(
    currentSettings: Settings,
    usageData: UsageData
  ): Optimization[];

  // Cross-platform preference sync
  syncPreferences(
    platforms: Platform[]
  ): Promise<void>;
}
```

Implementation Guidelines

Phase 1: Foundation (Months 1-3)

1. Core architecture and module system

2. Basic sensory management for web
3. Simple cognitive load indicators
4. Documentation framework
5. Initial React components

Phase 2: AI Integration (Months 4-6)

1. OpenAI/Anthropic API wrappers
2. Predictable AI behaviors
3. Explanation generation system
4. Attention management for chat interfaces
5. Python bindings for ML frameworks

Phase 3: VR/AR Support (Months 7-9)

1. Unity SDK development
2. WebXR implementation
3. Comfort zone system
4. Spatial audio management
5. Motion sickness prevention

Phase 4: Quantum & Advanced (Months 10-12)

1. Quantum visualization components
2. Circuit builder interfaces
3. Advanced testing suite
4. Analytics dashboard
5. Community contribution system

Usage Examples

Example 1: AI Chatbot Integration

```
import { NeuroAdapt } from '@neuroadapt/core';
import { AIAdapter } from '@neuroadapt/ai';

const neuro = new NeuroAdapt();
await neuro.loadUserProfile();
```

```javascript
const chatbot = new AIChatbot();
const adapter = new AIAdapter(chatbot);

adapter.configure({
  predictability: 'high',
  explanations: 'always',
  pacing: 'user-controlled',
  emotionalTone: 'calm-supportive'
});

// Automatically applies user's neurodiversity preferences
adapter.on('response', (response) => {
  neuro.adapt(response);
});
```

Example 2: VR Game Integration

```csharp
using NeuroAdapt.Unity;

public class AccessibleVRGame : MonoBehaviour {
    private NeuroAdaptManager neuroAdapt;

    void Start() {
        neuroAdapt = NeuroAdaptManager.Instance;
        neuroAdapt.InitializeWithProfile();

        // Automatically adjusts lighting, sounds, UI
        neuroAdapt.ApplyToScene(gameObject.scene);

        // Creates safe space portal
        neuroAdapt.CreateSafeSpace(transform.position);
    }
}
```

Example 3: Quantum Education App

```python
from neuroadapt import QuantumAdapter, CognitiveLoad
```

```
# Initialize with user profile
qa = QuantumAdapter(user_profile)

# Simplify quantum circuit visualization
circuit = QuantumCircuit(2)
circuit.h(0)
circuit.cx(0, 1)

# Adaptive visualization based on user needs
viz = qa.visualize(circuit,
    complexity='progressive',
    sensory_modes=['visual', 'audio'],
    pace='self-directed'
)

# Monitor cognitive load
with CognitiveLoad.monitor() as cl:
    if cl.is_high():
        viz.simplify()
        viz.add_break_suggestion()
```

Key Features Summary

1. Plug-and-Play Design

- Single line initialization
- Automatic preference detection
- Progressive enhancement approach
- Fallback mechanisms

2. Privacy-First

- Local preference storage
- Anonymized analytics
- User-controlled data sharing
- GDPR/CCPA compliant

3. Performance Optimized

- Lazy loading modules

- WebAssembly for intensive operations
- Efficient render pipelines
- Battery-conscious mobile features

4. **Developer Experience**

- Comprehensive TypeScript types
- Interactive documentation
- Code playground
- Migration tools
- CLI scaffolding

5. **Community-Driven**

- Open source (MIT license)
- Plugin ecosystem
- Community profiles marketplace
- Regular accessibility audits

Success Metrics

- **Adoption**: 10,000+ developers using SDK within Year 1
- **Coverage**: Support for 15+ neurodivergent profiles
- **Performance**: < 5% overhead on application performance
- **Satisfaction**: 4.5+ developer satisfaction rating
- **Impact**: 1M+ end users benefiting from SDK features

Additional Considerations

- Regular consultation with neurodivergent communities
- Partnerships with accessibility organizations
- Academic collaboration for research-backed features
- Industry standard compliance (WCAG, Section 508, etc.)
- Continuous user testing with diverse participants
- Regular updates for new platform features
- Extensive localization support
- Educational resources for developers

This SDK will democratize neurodiversity inclusion in emerging technologies, ensuring that the next generation of AI, quantum, and VR applications are accessible to all minds from day one.