NeXSheild: Comprehensive 120-Task Implementation Plan

Phase 1: Research & Foundation (Tasks 1-15)

Task 1: Market Research & Competitive Analysis

- Analyze CrowdStrike, SentinelOne, Microsoft Defender architecture
- Study Indian cybersecurity market requirements
- Research CERT-In guidelines and compliance requirements
- Document feature gap analysis

Task 2: Technology Stack Research

- Evaluate C++ frameworks (Boost, Poco, Qt)
- Compare database options (SQLite, PostgreSQL, TimescaleDB)
- Research real-time communication protocols (WebSocket, gRPC)
- Study cryptographic libraries (OpenSSL, LibreSSL)

Task 3: Zero Trust Architecture Design

- Design device identity verification system
- Plan continuous authentication mechanism
- Design micro-segmentation strategy
- Create least privilege access model

Task 4: Compliance Framework Research

- Map DPDPA 2023 requirements

- Study ISO 27001 controls
- Analyze NIST cybersecurity framework
- Research RBI cybersecurity guidelines

Task 5: System Architecture Specification

- Create component interaction diagrams
- Design data flow models
- Plan error handling strategy
- Design logging and audit systems

Task 6: Performance Requirements Definition

- Set CPU/Memory usage targets
- Define network bandwidth requirements
- Establish detection latency benchmarks
- Create scalability targets (10K+ endpoints)

Task 7: Security Protocol Design

- Design mTLS certificate hierarchy
- Plan key rotation strategy
- Design secure update mechanism
- Create intrusion prevention rules

Task 8: Database Architecture Design

- Design local SQLite schema
- Design central PostgreSQL schema
- Plan data partitioning strategy
- Design backup and recovery procedures

Task 9: API Specification

- Define REST API endpoints
- Design gRPC service definitions
- Create WebSocket event specifications
- Design API versioning strategy

Task 10: Development Environment Setup

- Set up C++ development environment (VS Code/CLion)
- Configure Python backend environment
- Set up React.js frontend environment
- Configure CI/CD pipeline (GitLab CI/Jenkins)

Task 11: Build System Configuration

- Configure CMake for C++ projects
- Set up Python virtual environments
- Configure Webpack for frontend
- Create Docker development containers

Task 12: Testing Framework Setup

- Configure Google Test for C++
- Set up pytest for Python backend
- Configure Jest for React frontend
- Set up performance testing tools

Task 13: Security Development Setup

- Configure static analysis tools (Clang-Tidy, SonarQube)
- Set up dependency vulnerability scanning
- Configure code signing certificates

- Set up secure development workstations

Task 14: Documentation System

- Set up Doxygen for C++ documentation
- Configure Sphinx for Python documentation
- Set up Storybook for UI components
- Create API documentation (OpenAPI/Swagger)

Task 15: Project Management Setup

- Configure Jira/Asana for task tracking
- Set up Confluence/wiki for documentation
- Configure Slack/Teams for communication
- Set up version control (Git) with branching strategy

Phase 2: Core Infrastructure (Tasks 16-35)

Task 16: Cryptographic Foundation Implementation

```
cpp

// File: security/CryptoManager.cpp
class CryptoManager {
    bool initializeHSMConnection();
    Certificate generateDeviceCertificate(string deviceId);
    bytes encryptData(bytes plaintext, bytes key);
    bytes decryptData(bytes ciphertext, bytes key);
    bool verifyDigitalSignature(bytes data, bytes signature, Certificate cert);
};
```

Task 17: Certificate Authority Setup

- Implement root CA generation
- Create intermediate CA hierarchy
- Implement certificate revocation (CRL/OCSP)
- Design certificate validation framework

Task 18: Mutual TLS Implementation

```cpp
// File: network/TLSManager.cpp
class TLSManager {
    SSL_CTX* createSSLContext(bool isServer);
    bool loadCertificates(SSL_CTX* ctx, string certPath, string keyPath);
    bool verifyClientCertificate(SSL* ssl);
    bool performHandshake(SSL* ssl);
};
```

Task 19: Local Database Engine

```cpp
// File: storage/LocalDatabase.cpp
class LocalDatabase {
    bool initializeWALMode();
    bool storeTelemetry(TelemetryData data);
    vector<TelemetryData> readTelemetry(time_t start, time_t end);
    bool enforceRetentionPolicy();
    bool performVacuum();
};
```

Task 20: Central Database Schema Implementation

```sql
-- File: database/schema.sql
CREATE TABLE endpoints (
```

```
    endpoint_id UUID PRIMARY KEY,
    device_name VARCHAR(255),
    os_type VARCHAR(50),
    os_version VARCHAR(100),
    agent_version VARCHAR(50),
    last_seen TIMESTAMP,
    certificate_thumbprint VARCHAR(64),
    risk_score INTEGER DEFAULT 0,
    compliance_status VARCHAR(20)
);

CREATE INDEX idx_endpoints_last_seen ON endpoints(last_seen);
CREATE INDEX idx_endpoints_risk_score ON endpoints(risk_score);
```

Task 21: Data Encryption Layer

- Implement AES-256-GCM for data at rest
- Design key derivation function (PBKDF2)
- Implement secure key storage
- Create key rotation scheduler

Task 22: Secure Communication Protocol

```
cpp

// File: network/SecureChannel.cpp
class SecureChannel {
    bool establishSecureConnection(string serverUrl);
    bool sendEncryptedMessage(bytes message);
    bytes receiveEncryptedMessage();
    bool validateMessageIntegrity(bytes message, bytes mac);
};
```

Task 23: Configuration Management System

```
cpp
```

```cpp
// File: config/ConfigManager.cpp
class ConfigManager {
    bool loadConfiguration(string configPath);
    string getConfigValue(string key);
    bool validateConfiguration();
    bool updateConfiguration(string key, string value);
};
```

Task 24: Logging and Audit System

```cpp
// File: logging/AuditLogger.cpp
class AuditLogger {
    bool logSecurityEvent(SecurityEvent event);
    bool logSystemEvent(SystemEvent event);
    vector<AuditRecord> queryAuditLogs(QueryFilter filter);
    bool exportAuditTrail(string filePath);
};
```

Task 25: Error Handling Framework

- Design exception hierarchy
- Implement graceful error recovery
- Create error reporting mechanism
- Design fault tolerance patterns

Task 26: Memory Management System

- Implement secure memory allocation
- Design buffer overflow protection
- Create memory sanitization
- Implement resource cleanup

Task 27: Threading and Concurrency

```cpp
// File: utils/ThreadManager.cpp
class ThreadManager {
    bool createWorkerThread(ThreadFunction func);
    bool manageThreadPool(size_t poolSize);
    bool implementMutexLocks();
    bool handleThreadExceptions();
};
```

Task 28: Performance Monitoring

- Implement real-time metrics collection
- Design performance counters
- Create resource usage monitoring
- Implement performance optimization

Task 29: Local Cache Management

```cpp
// File: storage/CacheManager.cpp
class CacheManager {
    bool storeInCache(string key, bytes data, time_t ttl);
    bytes getFromCache(string key);
    bool invalidateCache(string key);
    bool clearExpiredEntries();
};
```

Task 30: Network Resilience

- Implement retry mechanisms with exponential backoff
- Design circuit breaker pattern
- Create offline operation mode
- Implement data synchronization

Task 31: Security Policy Engine

```cpp
// File: security/PolicyEngine.cpp
class PolicyEngine {
    bool loadPolicies(vector<SecurityPolicy> policies);
    PolicyDecision evaluateAction(ActionRequest request);
    bool enforcePolicy(PolicyDecision decision);
    bool auditPolicyViolations();
};
```

Task 32: Update Framework Foundation

- Design delta update mechanism
- Implement rollback capability
- Create update verification system
- Design atomic update process

Task 33: Telemetry Collection Framework

```cpp
// File: telemetry/TelemetryCollector.cpp
class TelemetryCollector {
    bool collectProcessTelemetry();
    bool collectNetworkTelemetry();
    bool collectFileSystemTelemetry();
    bool collectSystemMetrics();
};
```

Task 34: Data Compression System

- Implement zstd compression
- Design efficient serialization
- Create batch processing
- Implement streaming compression

Task 35: Health Monitoring System

```cpp
// File: monitoring/HealthMonitor.cpp
class HealthMonitor {
    SystemHealth checkSystemHealth();
    bool performSelfDiagnostics();
    bool reportHealthStatus();
    bool triggerRecoveryActions();
};
```

Phase 3: Endpoint Agent Core (Tasks 36-65)

Task 36: Process Monitoring Engine (Windows)

```cpp
// File: monitoring/WindowsProcessMonitor.cpp
class WindowsProcessMonitor {
    bool initializeProcessTracking();
    ProcessInfo trackProcessCreation(DWORD processId);
    bool monitorProcessTermination(DWORD processId);
    vector<ProcessInfo> getRunningProcesses();
    bool detectProcessInjection(ProcessInfo process);
};
```

Task 37: Process Monitoring Engine (Linux)

```cpp
// File: monitoring/LinuxProcessMonitor.cpp
class LinuxProcessMonitor {
    bool monitorProcFilesystem();
    bool trackProcessFork();
    bool monitorProcessExecutions();
```

```
    bool detectAnomalousProcessBehavior();
};
```

Task 38: File System Monitor

```
cpp

// File: monitoring/FileSystemMonitor.cpp
class FileSystemMonitor {
    bool startFileWatching(vector<string> paths);
    FileEvent monitorFileChanges(string filePath);
    bool computeFileHashes(string filePath);
    bool detectSuspiciousFileActivity();
};
```

Task 39: Network Activity Monitor

```
cpp

// File: monitoring/NetworkMonitor.cpp
class NetworkMonitor {
    bool captureNetworkConnections();
    NetworkConnection analyzeConnection(ConnectionInfo info);
    bool detectPortScanning();
    bool monitorDNSQueries();
    bool detectDataExfiltration();
};
```

Task 40: Registry/Config Monitor (Windows)

```
cpp

// File: monitoring/RegistryMonitor.cpp
class RegistryMonitor {
    bool monitorRegistryChanges(vector<string> keys);
    RegistryEvent trackRegistryModification(string keyPath);
    bool detectPersistenceMechanisms();
    bool monitorAutoStartLocations();
};
```

Task 41: System Call Interceptor (Linux)

```cpp
// File: monitoring/SyscallMonitor.cpp
class SyscallMonitor {
    bool hookSystemCalls();
    SyscallInfo interceptSyscall(int syscallNumber);
    bool analyzeSyscallPatterns();
    bool detectSyscallAnomalies();
};
```

Task 42: Memory Scanner

```cpp
// File: scanning/MemoryScanner.cpp
class MemoryScanner {
    bool scanProcessMemory(DWORD processId);
    bool detectCodeInjection();
    bool findMaliciousPatterns(bytes memoryRegion);
    bool dumpSuspiciousMemory(string outputPath);
};
```

Task 43: Behavioral Analysis Engine

```cpp
// File: detection/BehaviorAnalyzer.cpp
class BehaviorAnalyzer {
    ThreatScore analyzeProcessBehavior(ProcessInfo process);
    bool detectLateralMovement(NetworkActivity activity);
    bool identifyPrivilegeEscalation();
    bool analyzeAttackChains(vector<SecurityEvent> events);
};
```

Task 44: Signature-Based Detection

```cpp
// File: detection/SignatureDetector.cpp
class SignatureDetector {
    bool loadYARARules(string rulesPath);
    DetectionResult scanFileWithYARA(string filePath);
    bool updateSignatureDatabase();
    bool performHeuristicAnalysis();
};
```

Task 45: Machine Learning Integration

```cpp
// File: detection/MLDetector.cpp
class MLDetector {
    bool loadMLModel(string modelPath);
    AnomalyScore predictAnomaly(FeatureVector features);
    bool retrainModel(TrainingData data);
    bool detectZeroDayThreats(SystemActivity activity);
};
```

Task 46: Real-time Alert System

```cpp
// File: alerting/AlertManager.cpp
class AlertManager {
    bool generateAlert(AlertSeverity severity, string description);
    bool correlateAlerts(vector<Alert> alerts);
    bool triggerIncidentResponse(Alert alert);
    bool notifySecurityTeam(Alert alert);
};
```

Task 47: Automated Response Engine

```cpp
// File: response/ResponseEngine.cpp
class ResponseEngine {
    bool quarantineFile(string filePath);
    bool terminateProcess(DWORD processId);
    bool isolateEndpoint();
    bool collectForensicData(Incident incident);
};
```

Task 48: Forensic Data Collector

```cpp
// File: forensics/ForensicCollector.cpp
class ForensicCollector {
    bool captureMemoryDump(DWORD processId);
    bool collectNetworkPackets();
    bool extractRegistryHives();
    bool preserveEvidence(Incident incident);
};
```

Task 49: Threat Intelligence Integration

```cpp
// File: intelligence/ThreatIntelClient.cpp
class ThreatIntelClient {
    bool queryIOC(string indicator);
    bool updateLocalIntelFeed();
    bool shareThreatIntelligence(ThreatInfo info);
    bool checkReputation(string hash, string ip);
};
```

Task 50: Local Rule Engine

```cpp
// File: detection/RuleEngine.cpp
class RuleEngine {
    bool loadDetectionRules(string rulesPath);
    vector<Detection> evaluateRules(SystemState state);
    bool updateRuleSet(string newRules);
    bool optimizeRuleEvaluation();
};
```

Task 51: Performance Optimizer

```cpp
// File: optimization/PerformanceOptimizer.cpp
class PerformanceOptimizer {
    bool monitorResourceUsage();
    bool adjustMonitoringIntensity();
    bool implementLazyEvaluation();
    bool optimizeMemoryUsage();
};
```

Task 52: Resource Governor

```cpp
// File: management/ResourceGovernor.cpp
class ResourceGovernor {
    bool enforceCPULimits();
    bool manageMemoryUsage();
    bool controlNetworkBandwidth();
    bool prioritizeCriticalTasks();
};
```

Task 53: Self-Protection Mechanism

```cpp
// File: security/SelfProtection.cpp
class SelfProtection {
    bool detectTampering();
    bool protectCriticalFiles();
    bool monitorAgentProcess();
    bool implementAntiKillMechanism();
};
```

Task 54: Compliance Checker

```cpp
// File: compliance/ComplianceChecker.cpp
class ComplianceChecker {
    bool checkCERTInRequirements();
    bool validateDPDPACompliance();
    bool generateComplianceReport();
    bool remediateComplianceIssues();
};
```

Task 55: Data Loss Prevention

```cpp
// File: prevention/DLPEngine.cpp
class DLPEngine {
    bool monitorSensitiveData();
    bool preventDataExfiltration();
    bool classifyDataSensitivity();
    bool enforceDataHandlingPolicies();
};
```

Task 56: Encryption Monitor

```cpp
// File: security/EncryptionMonitor.cpp
class EncryptionMonitor {
    bool verifyDiskEncryption();
    bool monitorKeyUsage();
    bool detectCryptoMining();
    bool validateTLSConnections();
};
```

Task 57: Vulnerability Scanner

```cpp
// File: scanning/VulnerabilityScanner.cpp
class VulnerabilityScanner {
    bool scanSystemVulnerabilities();
    bool checkPatchLevels();
    bool identifyMisconfigurations();
    bool prioritizeVulnerabilities();
};
```

Task 58: Application Control

```cpp
// File: control/ApplicationController.cpp
class ApplicationController {
    bool enforceWhitelistPolicy();
    bool blockUnauthorizedApplications();
    bool validateDigitalSignatures();
    bool monitorScriptExecution();
};
```

Task 59: Device Control

```cpp
// File: control/DeviceController.cpp
class DeviceController {
    bool monitorUSBDevices();
    bool enforceDevicePolicy();
    bool detectUnauthorizedDevices();
    bool logDeviceUsage();
};
```

Task 60: Network Firewall Integration

```cpp
// File: network/FirewallManager.cpp
class FirewallManager {
    bool createFirewallRules();
    bool blockMaliciousIPs();
    bool monitorNetworkTraffic();
    bool implementMicrosegmentation();
};
```

Task 61: Log Consolidation Engine

```cpp
// File: logging/LogConsolidator.cpp
class LogConsolidator {
    bool collectSystemLogs();
    bool parseLogEntries();
    bool correlateLogEvents();
    bool compressLogData();
};
```

Task 62: Incident Timeline Builder

```cpp
// File: forensics/IncidentTimeline.cpp
class IncidentTimeline {
    bool reconstructAttackTimeline();
    bool visualizeEventSequence();
    bool identifyRootCause();
    bool estimateImpactScope();
};
```

Task 63: Risk Scoring Engine

```cpp
// File: risk/RiskScorer.cpp
class RiskScorer {
    int calculateEndpointRisk();
    bool updateRiskScores();
    bool prioritizeRemediation();
    bool visualizeRiskHeatmap();
};
```

Task 64: Backup and Recovery

```cpp
// File: recovery/BackupManager.cpp
class BackupManager {
    bool backupConfiguration();
    bool restoreFromBackup();
    bool verifyBackupIntegrity();
    bool automateBackupProcess();
};
```

Task 65: Agent Self-Update System

```cpp
// File: update/AgentUpdater.cpp
class AgentUpdater {
    bool checkForUpdates();
    bool downloadUpdatePackage();
    bool verifyUpdateSignature();
    bool applyUpdateSafely();
};
```

Phase 4: Management Console Backend (Tasks 66-85)

Task 66: FastAPI Backend Setup

```python
# File: backend/main.py
from fastapi import FastAPI, Security
from fastapi.middleware.cors import CORSMiddleware

app = FastAPI(title="NeXSheild Management Console")

@app.get("/api/v1/health")
async def health_check():
    return {"status": "healthy"}
```

Task 67: Database Models Implementation

```python
# File: backend/models.py
from sqlalchemy import Column, String, Integer, DateTime, JSON
from sqlalchemy.ext.declarative import declarative_base
```

```
Base = declarative_base()


class Endpoint(Base):
    __tablename__ = "endpoints"
    endpoint_id = Column(String, primary_key=True)
    device_name = Column(String)
    os_version = Column(String)
    last_seen = Column(DateTime)
    risk_score = Column(Integer)
```

Task 68: Authentication System

```
python

# File: backend/auth.py
from fastapi.security import HTTPBearer, HTTPAuthorizationCredentials


security = HTTPBearer()


async def verify_token(credentials: HTTPAuthorizationCredentials):
    # Implement JWT validation
    pass
```

Task 69: Device Registration API

```
python

# File: backend/api/endpoints.py
@app.post("/api/v1/endpoints/register")
async def register_endpoint(device_info: DeviceInfo):
    # Validate device certificate
    # Generate unique endpoint ID
    # Store device information
    pass
```

Task 70: Telemetry Ingestion API

```
python
```

```
# File: backend/api/telemetry.py
@app.post("/api/v1/telemetry")
async def receive_telemetry(telemetry_data: TelemetryBatch):
    # Validate telemetry signature
    # Store in database
    # Trigger analysis
    pass
```

Task 71: Policy Management API

```
python

# File: backend/api/policies.py
@app.post("/api/v1/policies")
async def create_policy(policy: SecurityPolicy):
    # Validate policy syntax
    # Store policy
    # Distribute to endpoints
    pass
```

Task 72: Alert Management System

```
python

# File: backend/alerting/manager.py
class AlertManager:
    async def process_alert(self, alert: Alert):
        # Correlate with existing alerts
        # Update risk scores
        # Notify security team
        pass
```

Task 73: Real-time WebSocket Implementation

```
python

# File: backend/websocket/manager.py
class WebSocketManager:
    async def broadcast_alert(self, alert: Alert):
```

```
        # Send real-time alerts to dashboard
        pass
```

Task 74: Report Generation Engine

```
python

# File: backend/reporting/generator.py
class ReportGenerator:
    async def generate_compliance_report(self):
        # Create PDF reports
        # Export to various formats
        pass
```

Task 75: Data Analytics Engine

```
python

# File: backend/analytics/engine.py
class AnalyticsEngine:
    async def analyze_threat_patterns(self):
        # Use pandas for data analysis
        # Generate insights
        pass
```

Task 76: Backup API Implementation

```
python

# File: backend/api/backup.py
@app.post("/api/v1/backup")
async def create_backup():
    # Backup database
    # Upload to secure storage
    pass
```

Task 77: Audit Log API

```
python

# File: backend/api/audit.py
@app.get("/api/v1/audit")
async def get_audit_logs():
    # Retrieve audit trail
    # Apply filters
    pass
```

Task 78: Threat Intelligence API

```
python

# File: backend/api/threat_intel.py
@app.post("/api/v1/threat-intel")
async def add_ioc(ioc: IndicatorOfCompromise):
    # Validate IOC format
    # Distribute to endpoints
    pass
```

Task 79: Compliance Check API

```
python

# File: backend/api/compliance.py
@app.get("/api/v1/compliance")
async def check_compliance():
    # Run compliance checks
    # Generate reports
    pass
```

Task 80: System Health API

```
python
```

```python
# File: backend/api/health.py
@app.get("/api/v1/system-health")
async def system_health():
    # Check all components
    # Return health status
    pass
```

Task 81: User Management API

```python
# File: backend/api/users.py
@app.post("/api/v1/users")
async def create_user(user: UserCreate):
    # Implement RBAC
    # Validate permissions
    pass
```

Task 82: Notification System

```python
# File: backend/notifications/manager.py
class NotificationManager:
    async def send_alert_notification(self, alert: Alert):
        # Email notifications
        # Slack integration
        # SMS alerts
        pass
```

Task 83: Data Export API

```python
# File: backend/api/export.py
@app.post("/api/v1/export")
async def export_data(export_request: ExportRequest):
    # Export to CSV/JSON
```

```
    # Handle large datasets
    pass
```

Task 84: Search and Filter API

```
python

# File: backend/api/search.py
@app.get("/api/v1/search")
async def search_events(query: str):
    # Full-text search
    # Advanced filtering
    pass
```

Task 85: Batch Operations API

```
python

# File: backend/api/batch.py
@app.post("/api/v1/batch/actions")
async def execute_batch_actions(actions: BatchActionRequest):
    # Bulk operations
    # Progress tracking
    pass
```

Phase 5: Dashboard Frontend (Tasks 86-105)

Task 86: React Application Setup

```
javascript

// File: frontend/src/App.js
import React from 'react';
import { BrowserRouter as Router } from 'react-router-dom';

function App() {
  return (
```

```
    <Router>
      <div className="App">
        <Header />
        <Sidebar />
        <MainContent />
      </div>
    </Router>
  );
}
```

Task 87: Component Library Creation

```
javascript

// File: frontend/src/components/UI/Button.js
import React from 'react';
import './Button.css';

const Button = ({ variant, children, ...props }) => {
  return (
    <button className={`btn btn-${variant}`} {...props}>
      {children}
    </button>
  );
};
```

Task 88: Authentication Components

```
javascript

// File: frontend/src/components/Auth/Login.js
import React, { useState } from 'react';
import { useAuth } from '../../contexts/AuthContext';

const Login = () => {
  const [credentials, setCredentials] = useState({});
  const { login } = useAuth();

  return (
```

```
    <div className="login-container">
      <LoginForm onSubmit={login} />
    </div>
  );
};
```

Task 89: Main Dashboard Layout

```
javascript

// File: frontend/src/components/Dashboard/Dashboard.js
const Dashboard = () => {
  return (
    <div className="dashboard">
      <RiskOverview />
      <ThreatTimeline />
      <EndpointList />
      <AlertPanel />
    </div>
  );
};
```

Task 90: Real-time Threat Visualization

```
javascript

// File: frontend/src/components/Visualization/ThreatMap.js
import { Network } from 'vis-network';

const ThreatMap = ({ threats }) => {
  useEffect(() => {
    const network = new Network(container, data, options);
  }, [threats]);

  return <div id="threat-map" />;
};
```

Task 91: Risk Scoring Dashboard

```javascript
// File: frontend/src/components/Dashboard/RiskScore.js
const RiskScore = ({ endpoints }) => {
  const riskData = endpoints.map(e => ({
    name: e.device_name,
    score: e.risk_score
  }));

  return <BarChart data={riskData} />;
};
```

Task 92: Alert Management Interface

```javascript
// File: frontend/src/components/Alerts/AlertManager.js
const AlertManager = () => {
  const [alerts, setAlerts] = useState([]);
  const [filters, setFilters] = useState({});

  return (
    <div>
      <AlertFilters onFilter={setFilters} />
      <AlertList alerts={filteredAlerts} />
    </div>
  );
};
```

Task 93: Policy Management Interface

```javascript
// File: frontend/src/components/Policies/PolicyEditor.js
const PolicyEditor = ({ policy, onSave }) => {
  const [rules, setRules] = useState(policy.rules);
```

```
  return (
    <div className="policy-editor">
      <RuleBuilder rules={rules} onChange={setRules} />
      <button onClick={() => onSave(rules)}>Save Policy</button>
    </div>
  );
};
```

Task 94: Endpoint Management Interface

```
javascript

// File: frontend/src/components/Endpoints/EndpointManager.js
const EndpointManager = () => {
  const [endpoints, setEndpoints] = useState([]);
  const [selected, setSelected] = useState(null);

  return (
    <div className="endpoint-manager">
      <EndpointList endpoints={endpoints} onSelect={setSelected} />
      <EndpointDetails endpoint={selected} />
    </div>
  );
};
```

Task 95: Reporting Interface

```
javascript

// File: frontend/src/components/Reports/ReportGenerator.js
const ReportGenerator = () => {
  const [reportType, setReportType] = useState('compliance');
  const [dateRange, setDateRange] = useState({});

  return (
    <div>
      <ReportConfig
        type={reportType}
        dateRange={dateRange}
```

```
        onChange={updateConfig}
      />
      <ReportPreview config={config} />
    </div>
  );
};
```

Task 96: Real-time Monitoring Components

```
javascript

// File: frontend/src/components/Monitoring/LiveMetrics.js
const LiveMetrics = () => {
  const [metrics, setMetrics] = useState({});

  useEffect(() => {
    const ws = new WebSocket('/ws/metrics');
    ws.onmessage = (event) => {
      setMetrics(JSON.parse(event.data));
    };
  }, []);

  return <MetricsDisplay metrics={metrics} />;
};
```

Task 97: Compliance Dashboard

```
javascript

// File: frontend/src/components/Compliance/ComplianceDashboard.js
const ComplianceDashboard = () => {
  const [complianceData, setComplianceData] = useState({});

  return (
    <div>
      <ComplianceScore score={complianceData.overallScore} />
      <RequirementList requirements={complianceData.requirements} />
    </div>
```

```
  );
};
```

Task 98: Incident Investigation Interface

```
javascript

// File: frontend/src/components/Incidents/IncidentInvestigator.js
const IncidentInvestigator = ({ incidentId }) => {
  const [timeline, setTimeline] = useState([]);
  const [evidence, setEvidence] = useState([]);

  return (
    <div>
      <IncidentTimeline events={timeline} />
      <EvidenceCollector evidence={evidence} />
      <InvestigationNotes incidentId={incidentId} />
    </div>
  );
};
```

Task 99: Search and Filter Components

```
javascript

// File: frontend/src/components/Search/AdvancedSearch.js
const AdvancedSearch = ({ onSearch }) => {
  const [query, setQuery] = useState('');
  const [filters, setFilters] = useState({});

  return (
    <div className="advanced-search">
      <SearchBox value={query} onChange={setQuery} />
      <FilterPanel filters={filters} onChange={setFilters} />
      <SearchButton onClick={() => onSearch({ query, filters })} />
    </div>
  );
};
```

Task 100: Settings and Configuration

```
javascript

// File: frontend/src/components/Settings/SystemSettings.js
const SystemSettings = () => {
  const [settings, setSettings] = useState({});

  return (
    <div className="settings-panel">
      <NotificationSettings />
      <IntegrationSettings />
      <SecuritySettings />
      <BackupSettings />
    </div>
  );
};
```

Task 101: Responsive Design Implementation

- Implement mobile-first CSS
- Create responsive grid system
- Optimize for tablet devices
- Ensure touch-friendly interfaces

Task 102: Dark/Light Theme Implementation

```
javascript

// File: frontend/src/contexts/ThemeContext.js
const ThemeContext = createContext();

export const ThemeProvider = ({ children }) => {
  const [theme, setTheme] = useState('dark');

  return (
    <ThemeContext.Provider value={{ theme, setTheme }}>
      <div className={`theme-${theme}`}>
```

```
        {children}
      </div>
    </ThemeContext.Provider>
  );
};
```

Task 103: Performance Optimization

- Implement React.memo for components
- Use useCallback and useMemo hooks
- Implement virtual scrolling for large lists
- Optimize bundle size with code splitting

Task 104: Accessibility Implementation

- Add ARIA labels and roles
- Implement keyboard navigation
- Ensure screen reader compatibility
- Add high contrast mode

Task 105: Progressive Web App Features

- Implement service worker for offline functionality
- Add app manifest for installation
- Implement push notifications
- Create offline fallback pages

Phase 6: Advanced Features (Tasks 106-115)

Task 106: Machine Learning Integration

```
python
```

```
# File: backend/ml/anomaly_detector.py
class AnomalyDetector:
    def train_behavioral_model(self, training_data):
        # Use scikit-learn or TensorFlow
        pass

    def detect_anomalies(self, real_time_data):
        # Real-time anomaly detection
        pass
```

Task 107: Natural Language Processing

```
python

# File: backend/nlp/threat_analyzer.py
class ThreatAnalyzer:
    def analyze_threat_reports(self, text_data):
        # Extract IOCs from text
        # Classify threat severity
        pass
```

Task 108: Advanced Correlation Engine

```
python

# File: backend/correlation/engine.py
class CorrelationEngine:
    def correlate_events(self, events):
        # Identify attack patterns
        # Reduce false positives
        pass
```

Task 109: Predictive Analytics

```
python

# File: backend/analytics/predictive.py
class PredictiveAnalyzer:
    def predict_emerging_threats(self, historical_data):
```

```
        # Time series analysis
        # Threat forecasting
        pass
```

Task 110: Automation Playbooks

```
python

# File: backend/automation/playbook_engine.py
class PlaybookEngine:
    def execute_incident_response(self, incident):
        # Automated containment
        # Evidence collection
        pass
```

Task 111: API Gateway Implementation

```
python

# File: backend/gateway/main.py
class APIGateway:
    def rate_limit_requests(self):
        # Implement rate limiting
        pass

    def validate_api_keys(self):
        # API key management
        pass
```

Task 112: Microservices Architecture

- Decompose monolith into microservices
- Implement service discovery
- Configure API gateway routing
- Set up inter-service communication

Task 113: Containerization

```
dockerfile

# File: docker/Dockerfile.backend
FROM python:3.11-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY . .
CMD ["uvicorn", "main:app", "--host", "0.0.0.0"]
```

Task 114: Orchestration Setup

```
yaml

# File: kubernetes/deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nexsheild-backend
spec:
  replicas: 3
  template:
    spec:
      containers:
      - name: backend
        image: nexsheild/backend:latest
```

Task 115: Service Mesh Implementation

- Implement Istio for service mesh
- Configure traffic management
- Set up security policies
- Implement observability

Phase 7: Testing & Quality (Tasks 116-120)

Task 116: Comprehensive Test Suite

```cpp
// File: tests/unit/TestProcessMonitor.cpp
TEST_F(ProcessMonitorTest, DetectMaliciousProcess) {
    ProcessInfo maliciousProcess = createMockMaliciousProcess();
    EXPECT_TRUE(monitor.detectThreat(maliciousProcess));
}
```

Task 117: Performance Testing

- Load testing with 10,000 simulated endpoints
- Stress testing under high alert conditions
- Endurance testing for 72+ hours
- Scalability testing

Task 118: Security Testing

- Penetration testing by external team
- Vulnerability assessment
- Code review for security flaws
- Red team exercises

Task 119: User Acceptance Testing

- Pilot deployment with selected organizations
- Feedback collection and incorporation
- Usability testing
- Compliance validation

Task 120: Documentation Finalization

- User manuals and administration guides
- API documentation
- Deployment guides
- Troubleshooting documentation