AI Diagnostics System

Overview

The Sports Bar TV Controller includes a comprehensive AI-powered diagnostics system that allows both users and the AI assistant to monitor system health, identify issues, and troubleshoot problems automatically.

Diagnostic Features

1. AI Diagnostics Page (`/ai-diagnostics`)

A comprehensive dashboard that provides real-time system health monitoring with the following features:

- **Overall Health Score**: Percentage-based health metric (0-100%)
- **System Status**: Visual indicators for healthy, warning, and error states
- **Component Checks**: Individual health checks for:
- Bartender Remote Control
- Device Mapping
- AI Providers
- Device AI Analysis
- **Quick Actions**: Links to AI Hub, System Admin, and Device Config
- **Real-time Updates**: Refresh button to re-run diagnostics on demand

Access: Navigate to `/ai-diagnostics` or click the "AI System Diagnostics" link in the AI Hub Configuration tab.

2. AI Diagnostic Runner API (`/api/ai/run-diagnostics`)

A powerful API endpoint that the AI assistant can use to run comprehensive system diagnostics programmatically.

Available Diagnostic Checks:

1. **Database Connectivity** (`database`)
   - Tests database connection
   - Verifies query execution
   - Status: pass/fail

2. **AI Providers** (`ai_providers`)
   - Counts active vs total AI providers
   - Checks provider configuration
   - Status: pass (>0 active) / warning (0 active) / fail (error)

3. **Matrix Inputs** (`matrix_inputs`)
   - Checks Wolf Pack matrix input configuration
   - Counts active inputs
   - Status: pass (>0 active) / warning (0 active) / fail (error)

4. **IR Devices** (`ir_devices`)
   - Validates IR device configuration file
   - Counts active IR devices
   - Status: pass (>0 active) / warning (0 active or file issues)

5. **Device Mapping** (`device_mapping`)
   - Analyzes matrix input to IR device mapping
   - Calculates mapping percentage
   - Status: pass (≥80%) / warning (50-79%) / fail (<50%)

6. **Knowledge Base** (`knowledge_base`)
   - Checks AI knowledge base directory
   - Counts documents
   - Status: pass (>0 docs) / warning (0 docs or missing)

7. **System Logs** (`system_logs`)
   - Analyzes recent system logs
   - Calculates error rate
   - Status: pass (<5% errors) / warning (5-15%) / fail (>15%)

8. **API Health** (`api_health`)
   - Tests critical API endpoints
   - Checks response status
   - Status: pass (all responding) / warning (some responding) / fail (most failing)

API Usage:

POST Request:

```
{
  "checks": ["all"],  // or specific checks: ["database", "ai_providers"]
  "detailed": false   // set to true for detailed information
}
```

Response:

```
{
  "success": true,
  "overallHealth": "healthy",
  "healthScore": 87,
  "summary": {
    "total": 8,
    "passed": 7,
    "failed": 0,
    "warnings": 1
  },
  "diagnostics": [
    {
      "name": "Database Connectivity",
      "category": "infrastructure",
      "status": "pass",
      "message": "Database connection is healthy",
      "timestamp": "2025-10-05T10:30:00.000Z"
    }
    // ... more checks
  ],
  "executionTime": "1234ms",
  "timestamp": "2025-10-05T10:30:00.000Z"
}
```

GET Request (Quick Check):

```
GET /api/ai/run-diagnostics?quick=true
```

Returns a quick health check of database and AI providers only.

3. AI Providers Status API (`/api/ai-providers/status`)

Dedicated endpoint for checking AI provider status.

GET Request:

```
GET /api/ai-providers/status
```

Response:

```json
{
  "success": true,
  "active": 2,
  "total": 3,
  "health": "healthy",
  "healthScore": 67,
  "providers": [
    {
      "id": "1",
      "name": "OpenAI",
      "type": "openai",
      "isActive": true,
      "priority": 1,
      "successRate": 98.5,
      "totalRequests": 1234,
      "failedRequests": 18
    }
    // ... more providers
  ],
  "timestamp": "2025-10-05T10:30:00.000Z"
}
```

POST Actions:
- `test_provider` : Test a specific AI provider
- `refresh_status` : Refresh provider status

4. Intelligent Troubleshooter Component

AI-powered device diagnostics and automated repair system.

Features:
- Progressive diagnostic scanning
- Issue classification (connection, performance, configuration, hardware)
- Severity levels (low, medium, high, critical)
- Recommended actions with success probability
- One-click automated fixes
- Real-time progress tracking

Location: Available in AI Hub under “AI Tools” tab

5. Device AI Analysis API (`/api/devices/ai-analysis`)

Provides AI-powered insights and recommendations for devices.

Features:
- Device health metrics
- Performance analysis
- Usage pattern detection
- Optimization recommendations
- Predictive analytics

6. Bartender Remote Diagnostics (`/api/diagnostics/bartender-remote`)

Specialized diagnostics for the bartender remote control system.

Checks:
- Matrix input configuration
- IR device mapping
- Fallback data usage
- Component filtering logic

7. Device Mapping Diagnostics (`/api/diagnostics/device-mapping`)

Analyzes the mapping between Wolf Pack inputs and IR devices.

Provides:
- Mapping status for each input
- Control readiness assessment
- Configuration recommendations
- Summary statistics

How the AI Can Use Diagnostics

1. Proactive Monitoring

The AI assistant can periodically run diagnostics to monitor system health:

```
// Run all diagnostics
const response = await fetch('/api/ai/run-diagnostics', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({ checks: ['all'], detailed: true })
})
const results = await response.json()
```

2. Targeted Troubleshooting

When a user reports an issue, the AI can run specific diagnostic checks:

```
// Check only device-related components
const response = await fetch('/api/ai/run-diagnostics', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
    checks: ['matrix_inputs', 'ir_devices', 'device_mapping'],
    detailed: true
  })
})
```

3. Quick Health Checks

For rapid status verification:

```
// Quick check
const response = await fetch('/api/ai/run-diagnostics?quick=true')
const quickStatus = await response.json()
```

4. Automated Fixes

After identifying issues, the AI can execute automated fixes:

```
// Execute a fix
const response = await fetch('/api/devices/execute-fix', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
    actionId: 'action_1',
    deviceId: 'device_123'
  })
})
```

User Access

Via Web Interface

1. **AI Hub** → Configuration Tab → “AI System Diagnostics” link
2. Direct URL: `http://localhost:3000/ai-diagnostics`
3. **AI Hub** → AI Tools Tab → Intelligent Troubleshooter

Via AI Assistant

Users can ask the AI assistant to run diagnostics:

- “Run system diagnostics”
- “Check the health of the AI system”
- “Diagnose any issues with the devices”
- “What’s the status of the AI providers?”
- “Check if all devices are properly mapped”

The AI will automatically use the diagnostic APIs to gather information and provide detailed reports.

Diagnostic Categories

Infrastructure

- Database connectivity
- API endpoint health
- Network connectivity

AI System

- AI provider status
- Knowledge base health
- Model availability

Hardware

- Matrix input configuration
- IR device status
- Device connectivity

Configuration

- Device mapping
- System settings
- Integration setup

Monitoring

- System logs analysis
- Error rate tracking
- Performance metrics

Best Practices

For Users

1. **Regular Checks**: Run diagnostics weekly or after system changes
2. **Before Events**: Check system health before major sporting events
3. **After Issues**: Run diagnostics immediately after experiencing problems
4. **Review Details**: Enable detailed mode for thorough analysis

For AI Assistant

1. **Proactive Monitoring**: Run diagnostics during idle periods
2. **Context-Aware**: Choose specific checks based on user's issue
3. **Detailed Analysis**: Always request detailed information for troubleshooting
4. **Follow-up**: Re-run diagnostics after applying fixes to verify success
5. **Log Results**: Keep track of diagnostic results for pattern analysis

Troubleshooting Common Issues

All Diagnostics Failing

- Check database connection
- Verify server is running
- Check network connectivity

AI Providers Warning

- Verify API keys are configured
- Check provider service status
- Review provider priority settings

Device Mapping Issues

- Verify matrix inputs are configured
- Check IR devices JSON file exists
- Ensure input channels match device assignments

High Error Rate in Logs

- Review recent system logs
- Check for recurring error patterns
- Investigate failed operations

Future Enhancements

Planned improvements to the diagnostic system:

1. **Automated Scheduling**: Periodic automatic diagnostic runs

2. **Alert System**: Notifications when health score drops below threshold
3. **Historical Tracking**: Trend analysis of system health over time
4. **Predictive Diagnostics**: AI-powered prediction of potential issues
5. **Self-Healing**: Automatic execution of fixes for common issues
6. **Performance Benchmarking**: Compare current performance to baselines
7. **Integration Testing**: Automated testing of external integrations

API Reference Summary

Endpoint	Method	Purpose
`/api/ai/run-diagnostics`	POST	Run comprehensive diagnostics
`/api/ai/run-diagnostics?quick=true`	GET	Quick health check
`/api/ai-providers/status`	GET	Check AI provider status
`/api/diagnostics/bartender-remote`	GET	Bartender remote diagnostics
`/api/diagnostics/device-mapping`	GET	Device mapping analysis
`/api/devices/ai-analysis`	GET	Device AI insights
`/api/devices/intelligent-diagnostics`	GET	Intelligent device diagnostics
`/api/devices/execute-fix`	POST	Execute automated fix

Conclusion

The AI Diagnostics System provides comprehensive monitoring and troubleshooting capabilities for the Sports Bar TV Controller. It enables both users and the AI assistant to maintain system health, identify issues quickly, and apply fixes automatically. The system is designed to be proactive, intelligent, and user-friendly, ensuring optimal performance of the entire sports bar control system.