Diagnostics System Deployment Guide

🚀 Quick Start

The AI diagnostics system is now ready to deploy. Follow these steps to get it running on the production server.

📋 Prerequisites

- Node.js and npm installed
- PM2 installed globally (`npm install -g pm2`)
- Access to the production server (135.131.39.26:223)
- Database already set up and running

🔧 Installation Steps

1. Pull Latest Changes

```
cd /home/ubuntu/Sports-Bar-TV-Controller
git pull origin diagnostics-daemon
```

2. Install Dependencies

```
npm install
```

The following packages are required and should be installed:
- `axios` - For HTTP requests
- `node-cron` - For scheduling

3. Update Database Schema

```
npx prisma generate
npx prisma db push
```

This will add the following new models:
- SystemHealthCheck
- Issue
- Fix
- SystemMetric
- LearningPattern
- DiagnosticRun

4. Test the System

```
node scripts/diagnostics/test-diagnostics.js
```

You should see:

```
🎉 All tests passed! Diagnostics system is ready.
```

5. Start the Diagnostics Scheduler

```
pm2 start scripts/diagnostics/scheduler.js --name diagnostics-scheduler
pm2 save
```

6. Verify It's Running

```
pm2 status
pm2 logs diagnostics-scheduler
```

📊 What Gets Monitored

Light Checks (Every 5 Minutes)

- ✅ PM2 process health
- ✅ API endpoint availability (/api/health)
- ✅ Database connectivity
- ✅ Disk space (warns at 80%, critical at 90%)
- ✅ Memory usage (warns at 85%, critical at 95%)
- ✅ System load average

Deep Diagnostics (Sunday 5:00 AM)

- 🔍 Full dependency audit
- 🔒 Security vulnerability scan
- ⚡ Performance analysis (7-day trends)
- 📋 Log file analysis
- 🗄️ Database integrity check
- 🔌 External integration testing
- ⚙️ Configuration validation
- 💡 Optimization recommendations

Self-Healing (Automatic)

- 🔄 Restart crashed PM2 processes
- 🧹 Clean disk space when >90%
- 💾 Handle high memory usage
- 📦 Reinstall missing dependencies
- 🗄️ Repair corrupted database
- 📋 Rotate large log files

🌐 API Endpoints

Once deployed, you can manually trigger diagnostics via API:

Run Light Check

```
curl -X POST http://192.168.1.25:3000/api/diagnostics/light-check
```

Run Deep Diagnostics

```
curl -X POST http://192.168.1.25:3000/api/diagnostics/deep
```

Trigger Self-Healing

```
curl -X POST http://192.168.1.25:3000/api/diagnostics/self-heal
```

Get Status

```
curl http://192.168.1.25:3000/api/diagnostics/status
```

📈 Viewing Results

Database Queries

You can query the diagnostics data using Prisma:

```
// Get recent health checks
const checks = await prisma.systemHealthCheck.findMany({
  take: 50,
  orderBy: { timestamp: 'desc' }
});

// Get open issues
const issues = await prisma.issue.findMany({
  where: { status: 'open' },
  orderBy: { severity: 'desc' }
});

// Get recent diagnostic runs
const runs = await prisma.diagnosticRun.findMany({
  take: 10,
  orderBy: { timestamp: 'desc' }
});
```

Log Files

Check the PM2 logs for diagnostics output:

```
pm2 logs diagnostics-scheduler
pm2 logs diagnostics-scheduler --lines 100
```

🔧 Configuration

Adjust Thresholds

Edit the CONFIG object in each script:

light-check.js:

```
const CONFIG = {
  DISK_WARNING_THRESHOLD: 80,      // percent
  DISK_CRITICAL_THRESHOLD: 90,     // percent
  MEMORY_WARNING_THRESHOLD: 85,    // percent
  MEMORY_CRITICAL_THRESHOLD: 95,   // percent
  API_TIMEOUT: 5000,               // milliseconds
};
```

deep-diagnostics.js:

```
const CONFIG = {
  MAX_LOG_SIZE: 100 * 1024 * 1024, // 100MB
  DAYS_TO_ANALYZE: 7               // days
};
```

Change Schedule

Edit **scheduler.js** to change the schedule:

```
// Light check - currently every 5 minutes
const lightCheckJob = cron.schedule('*/5 * * * *', ...);

// Deep diagnostics - currently Sunday 5:00 AM
const deepDiagnosticsJob = cron.schedule('0 5 * * 0', ...);
```

🐛 Troubleshooting

Scheduler Not Running

```
# Check status
pm2 status

# Restart
pm2 restart diagnostics-scheduler

# View logs
pm2 logs diagnostics-scheduler --lines 50
```

Database Connection Issues

```
# Check database file
ls -lh /home/ubuntu/Sports-Bar-TV-Controller/prisma/data/sports_bar.db

# Test connection
node -e "const { PrismaClient } = require('@prisma/client'); const prisma = new Pris-
maClient(); prisma.\$queryRaw\`SELECT 1\`.then(() => con-
sole.log('OK')).catch(console.error);"
```

Permission Issues

```
# Make scripts executable
chmod +x scripts/diagnostics/*.js

# Check ownership
ls -la scripts/diagnostics/
```

PM2 Not Found

```
# Install PM2 globally
npm install -g pm2

# Or use npx
npx pm2 start scripts/diagnostics/scheduler.js --name diagnostics-scheduler
```

📊 Expected Behavior

First Run

- Light check runs immediately on scheduler start
- Creates initial database records
- May detect issues (PM2 not running, API down, etc.)
- Self-healing attempts to fix detected issues

Ongoing Operation

- Light checks run every 5 minutes
- Issues are logged to database
- Self-healing triggers automatically for fixable issues
- Deep diagnostics run every Sunday at 5:00 AM
- All results stored in database for analysis

Notifications

Currently, the system logs to:
- PM2 logs (`pm2 logs diagnostics-scheduler`)
- Database (query via Prisma or API)
- Console output (when run manually)

Future enhancements can add:
- Email notifications
- Slack/Discord webhooks
- SMS alerts for critical issues

🎯 Success Criteria

After deployment, verify:

1. ✅ Scheduler is running in PM2
2. ✅ Light checks execute every 5 minutes
3. ✅ Database records are being created

4. ✅ API endpoints respond correctly
5. ✅ Self-healing triggers on issues
6. ✅ No errors in PM2 logs

📚 Documentation

- Full documentation: `/docs/diagnostics-system.md`
- Script README: `/scripts/diagnostics/README.md`
- System profile: `/home/ubuntu/system-profile.md`

🔄 Maintenance

Weekly

- Review deep diagnostics report (Monday morning)
- Check for recurring issues
- Implement optimization recommendations

Monthly

- Review learning patterns
- Adjust thresholds if needed
- Clean old diagnostic data (automatic)

As Needed

- Update dependencies
- Add new monitoring points
- Enhance self-healing capabilities

🚨 Emergency Procedures

Stop Diagnostics

```
pm2 stop diagnostics-scheduler
```

Disable Self-Healing

Comment out the trigger in `light-check.js`:

```
// await this.triggerSelfHealing();
```

Reset Database

```
# Backup first!
cp prisma/data/sports_bar.db prisma/data/sports_bar.db.backup

# Reset diagnostics data
node -e "const { PrismaClient } = require('@prisma/client'); const prisma = new Pris-
maClient(); Promise.all([prisma.systemHealthCheck.deleteMany(),
prisma.issue.deleteMany(), prisma.diagnosticRun.deleteMany()]).then(() =>
console.log('Reset complete'));"
```

📞 Support

For issues or questions:
1. Check PM2 logs: `pm2 logs diagnostics-scheduler`
2. Review database: Query `DiagnosticRun` table
3. Run manual tests: `node scripts/diagnostics/test-diagnostics.js`
4. Check documentation: `/docs/diagnostics-system.md`

Ready to deploy! 🚀

The system is fully tested and ready for production use. Follow the installation steps above to get it running on the server.