Sports Bar TV Controller - Complete System Documentation

🔐 SSH Server Access Credentials

IMPORTANT: Server Access Information

Parameter	Value
Server	24.123.87.42
Port	224
Username	ubuntu
Password	6809233DjD$$$ (THREE dollar signs)
Authentication	Password only (no SSH key required)
Project Path	~/Sports-Bar-TV-Controller
Application URL	http://24.123.87.42:3001
GitHub Repository	https://github.com/dfultonthebar/Sports-Bar-TV-Controller

Connection Command:

```
ssh -p 224 ubuntu@24.123.87.42
```

Common SSH Operations:

```
# Connect to server
ssh -p 224 ubuntu@24.123.87.42

# Copy files to server (SCP)
scp -P 224 localfile.txt ubuntu@24.123.87.42:~/destination/

# Copy files from server
scp -P 224 ubuntu@24.123.87.42:~/remote/file.txt ./local/

# SSH with port forwarding (for local development)
ssh -p 224 -L 3001:localhost:3001 ubuntu@24.123.87.42
```

Standard Deployment Workflow:

1. SSH into server: `ssh -p 224 ubuntu@24.123.87.42`

2. Navigate to project: `cd ~/Sports-Bar-TV-Controller`
3. Pull latest changes: `git pull origin main`
4. Install dependencies: `npm install`
5. Build application: `npm run build`
6. Restart PM2: `pm2 restart sports-bar-tv-controller`
7. Check logs: `pm2 logs sports-bar-tv-controller`

Security Notes:
- SSH is configured on non-standard port 224 for additional security
- Password authentication is enabled (no SSH key required)
- Ensure password is kept secure and not shared publicly
- Consider implementing SSH key authentication for enhanced security in future

---

📑 Table of Contents

System Overview

The Sports Bar TV Controller is a comprehensive web application designed to manage TV displays, matrix video routing, and sports content scheduling for sports bar environments. The system integrates with Wolfpack matrix switchers and provides automated scheduling capabilities.

Key Features

- **Matrix Video Routing**: Control Wolfpack HDMI matrix switchers
- **TV Schedule Management**: Automated daily on/off scheduling with selective TV control
- **Sports Content Guide**: Display and manage sports programming
- **Multi-Zone Audio**: Control audio routing for different zones (Atlas audio processor)
- **Web-Based Interface**: Responsive UI accessible from any device
- **AI-Powered Assistant**: Context-aware assistance for troubleshooting and system management
- **Automated Backup System**: Daily backups with configurable retention
- **Real-Time Diagnostics**: Comprehensive system health monitoring

Technology Stack

- **Frontend**: Next.js 14, React, TypeScript, Tailwind CSS
- **Backend**: Next.js API Routes, Prisma ORM
- **Database**: PostgreSQL (primary), SQLite (development)
- **Hardware Integration**:
- Wolfpack HDMI Matrix Switchers (via HTTP API)
- Atlas IED Audio Processors (via HTTP API)
- DirecTV Receivers (via HTTP API)
- Fire TV devices (via ADB)
- **AI/ML**: Ollama (local AI), Knowledge Base System
- **Process Management**: PM2

Architecture

Application Structure

```
Sports-Bar-TV-Controller/
├── src/
│   ├── app/                    # Next.js app router pages
│   │   ├── ai-hub/             # AI features hub
│   │   ├── ai-diagnostics/     # System diagnostics
│   │   ├── audio-control/      # Audio zone management
│   │   ├── backup-restore/     # Backup UI
│   │   └── system-admin/       # System administration
│   ├── components/             # React components
│   │   ├── MatrixControl.tsx   # Matrix output controls
│   │   ├── AudioZoneControl.tsx # Audio zone controls
│   │   ├── SportsGuide.tsx     # Sports programming guide
│   │   ├── ApiKeysManager.tsx  # API configuration
│   │   └── tv-guide/           # TV guide components
│   ├── lib/                    # Utility libraries
│   │   ├── prisma.ts           # Prisma client singleton
│   │   ├── wolfpack.ts         # Wolfpack API client
│   │   ├── cache-service.ts    # Caching system
│   │   ├── ai-knowledge.ts     # AI knowledge base
│   │   └── sportsGuideApi.ts   # Sports guide API client
│   └── pages/api/              # API routes
│       ├── matrix/             # Matrix control endpoints
│       ├── wolfpack/           # Wolfpack integration
│       ├── atlas/              # Atlas audio processor
│       ├── ai/                 # AI features endpoints
│       ├── backup/             # Backup/restore endpoints
│       └── schedule/           # Scheduling endpoints
├── prisma/
│   └── schema.prisma           # Database schema
├── config/                     # Configuration files
│   ├── *.template.json         # Configuration templates
│   └── *.local.json            # Local configuration (gitignored)
├── data/                       # Data files
│   ├── ai-knowledge-base.json  # AI knowledge base
│   └── atlas-configs/          # Atlas configurations
├── scripts/                    # Utility scripts
│   ├── build-knowledge-base.ts # KB builder
│   └── init-local-config.sh    # Config initializer
└── public/                     # Static assets
```

Component Architecture

Matrix Control System

The matrix control system manages video routing between sources and displays:

- **Outputs 1-4**: TV 01-04 (Full matrix outputs with all controls)
- Power on/off controls
- Active status checkbox
- Source routing buttons
- Audio output configuration
- Full Wolfpack integration
- **Outputs 5-32**: Regular matrix outputs (Full controls)

- All features of outputs 1-4
- Configurable labels and settings

- **Outputs 33-36**: Matrix 1-4 Audio (Special audio routing outputs)

- Video input selection for audio
- Dynamic labels based on selected input
- Integration with Atlas audio processor

Audio Control System

The Atlas audio processor integration provides:

- **Zone Control**: Independent audio zones with volume and source control
- **Dynamic Labeling**: Zone labels reflect selected video inputs
- **Input Management**: Configure audio inputs with gain, EQ, and routing
- **Output Management**: Configure speakers with delay, limiting, and grouping
- **Scene Management**: Save and recall audio configurations

TV Selection System

Allows granular control over which TVs participate in automated schedules:

- **dailyTurnOn**: Boolean flag indicating if TV should turn on during morning schedule
- **dailyTurnOff**: Boolean flag indicating if TV responds to “all off” command
- Configured per output in the database
- Accessible via `/api/matrix/outputs-schedule` endpoint

Database Schema

Key Models

MatrixOutput

Represents a video output (TV display):

```
model MatrixOutput {
  id               Int      @id @default(autoincrement())
  outputNumber     Int      @unique
  label            String
  enabled          Boolean  @default(true)
  isActive         Boolean  @default(false)
  currentInput     Int?
  audioOutput      Int?
  resolution       String?
  dailyTurnOn      Boolean  @default(true)    # Morning schedule participation
  dailyTurnOff     Boolean  @default(true)    # "All off" command participation
  isMatrixOutput   Boolean  @default(true)    # Differentiates simple vs matrix outputs
  createdAt        DateTime @default(now())
  updatedAt        DateTime @updatedAt
}
```

MatrixInput

Represents a video source:

```
model MatrixInput {
  id           Int      @id @default(autoincrement())
  inputNumber  Int      @unique
  label        String
  enabled      Boolean  @default(true)
  createdAt    DateTime @default(now())
  updatedAt    DateTime @updatedAt
}
```

WolfpackConfig

Stores Wolfpack matrix switcher configuration:

```
model WolfpackConfig {
  id         Int      @id @default(cuid())
  ipAddress  String   @unique
  name       String?
  createdAt  DateTime @default(now())
  updatedAt  DateTime @updatedAt
}
```

AudioProcessor

Stores Atlas audio processor configuration:

```
model AudioProcessor {
  id          String   @id @default(cuid())
  name        String
  ipAddress   String
  port        Int      @default(80)
  model       String   # AZM4, AZM8, etc.
  isActive    Boolean  @default(true)
  username    String?
  password    String?
  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt
}
```

QAEntry

Stores Q&A training data for AI:

```
model QAEntry {
  id           String   @id @default(cuid())
  question     String
  answer       String   @db.Text
  category     String   # system, api, features, configuration, troubleshooting, gen-
eral
  tags         String?  # JSON array
  sourceType   String   # manual, auto-generated, uploaded
  sourceFile   String?
  confidence   Float?   # 0.0-1.0 for auto-generated
  isActive     Boolean  @default(true)
  usageCount   Int      @default(0)
  lastUsed     DateTime?
  createdAt    DateTime @default(now())
  updatedAt    DateTime @updatedAt
}
```

TestLog

Stores test results for diagnostics:

```
model TestLog {
  id            String   @id @default(cuid())
  testType      String
  testName      String
  status        String
  inputChannel  Int?
  outputChannel Int?
  command       String?
  response      String?
  errorMessage  String?
  duration      Int?
  timestamp     DateTime @default(now())
  metadata      String?  # JSON
}
```

API Endpoints

Matrix Control

GET/POST `/api/matrix/outputs`

- **GET**: Retrieve all matrix outputs
- **POST**: Update output configuration
- **Body**: `{ outputNumber, label, enabled, dailyTurnOn, dailyTurnOff }`

GET `/api/matrix/outputs-schedule`

Retrieve outputs with schedule participation flags:

```json
{
  "outputs": [
    {
      "outputNumber": 33,
      "label": "Main Bar TV",
      "dailyTurnOn": true,
      "dailyTurnOff": true
    }
  ]
}
```

POST `/api/matrix/route`

Route a source to an output:

```json
{
  "input": 5,
  "output": 33
}
```

POST `/api/matrix/power`

Control output power:

```
{
  "output": 33,
  "state": "on"  // or "off"
}
```

GET/POST `/api/matrix/video-input-selection`

Get or set video input for Matrix 1-4 audio outputs:

```
{
  "outputNumber": 33,
  "videoInputNumber": 5
}
```

Wolfpack Integration

POST `/api/wolfpack/test-connection`

Test connectivity to Wolfpack matrix:

```
{
  "ipAddress": "192.168.1.100"
}
```

Response:

```
{
  "success": true,
  "message": "Successfully connected to Wolfpack matrix"
}
```

POST `/api/wolfpack/test-switching`

Test matrix switching functionality:

```
{
  "ipAddress": "192.168.1.100"
}
```

Atlas Audio Processor

GET `/api/audio-processor`

List all configured audio processors

POST `/api/atlas/upload-config`

Upload Atlas configuration:

```
{
  "processorId": "cmgjxa5ai...",
  "config": { ... }  // Full configuration object
}
```

GET `/api/atlas/download-config`

Download saved Atlas configuration:

```
GET /api/atlas/download-config?processorId=cmgjxa5ai...
```

GET `/api/atlas/ai-analysis`

Get AI analysis of Atlas audio system:

```
GET /api/atlas/ai-analysis?processorId=cmgjxa5ai...
```

Backup & Restore

GET `/api/backup`

List all available backups:

```
{
  "backups": [
    {
      "filename": "config-backup-20251010-120000.tar.gz",
      "size": 1024000,
      "timestamp": "2025-10-10T12:00:00.000Z"
    }
  ]
}
```

POST `/api/backup`

Create, restore, or delete backups:

```
// Create backup
{ "action": "create" }

// Restore backup
{ "action": "restore", "filename": "config-backup-..." }

// Delete backup
{ "action": "delete", "filename": "config-backup-..." }
```

AI Features

POST `/api/ai/enhanced-chat`

Chat with AI assistant using knowledge base:

```
{
  "message": "How do I configure the Atlas audio processor?",
  "useKnowledgeBase": true,
  "model": "llama3.2:3b"
}
```

POST `/api/ai/run-diagnostics`

Run system diagnostics:

```
{
  "checks": ["all"],  // or specific: ["database", "ai_providers"]
  "detailed": false
}
```

GET `/api/ai-providers/status`

Get AI provider status and statistics

GET `/api/ai/qa-entries`

List Q&A training entries (with filtering and search)

POST `/api/ai/qa-generate`

Generate Q&A pairs from documentation:

```
{
  "sourceType": "repository",  // or "documentation", "codebase"
  "model": "llama3.2:3b"
}
```

Sports Guide

GET `/api/sports-guide/status`

Get Sports Guide API configuration status

GET `/api/sports-guide/channels`

Fetch channel guide data with filtering

POST `/api/sports-guide/verify-key`

Verify Sports Guide API key

Channel Guide

GET `/api/channel-guide`

Get channel guide data (cable, satellite, streaming)

Device Subscriptions

POST `/api/device-subscriptions/poll`

Poll real device subscriptions (DirecTV, Fire TV)

Configuration Management

Wolfpack Matrix Configuration

Initial Setup

1. Navigate to **System Admin → Wolfpack Configuration**
2. Enter matrix IP address (e.g., `192.168.1.100`)
3. Click **Test Connection** to verify connectivity
4. Click **Test Switching** to verify control functionality
5. Save configuration

Input Configuration

1. Navigate to **System Admin → Matrix Inputs**
2. For each input (1-32):
   - Set descriptive label (e.g., "Cable Box 1", "Apple TV")
   - Enable/disable as needed
3. Save changes

Output Configuration

1. Navigate to **System Admin → Matrix Outputs**
2. For each output:
   - Set descriptive label (e.g., "Main Bar TV 1")
   - Enable/disable as needed
   - Set `dailyTurnOn` (participate in morning schedule)
   - Set `dailyTurnOff` (respond to "all off" command)
3. Save changes

Local Configuration System

A robust local configuration management system preserves system-specific settings when updating from GitHub.

Configuration Files

Template Files (Tracked in Git):
- `config/local.template.json` - System settings template
- `config/devices.template.json` - Device inventory template
- `config/sports-teams.template.json` - Team monitoring template

Local Files (Gitignored):
- `config/local.local.json` - Your system settings
- `config/devices.local.json` - Your device inventory
- `config/sports-teams.local.json` - Your team preferences

These local files are automatically created from templates and NEVER committed to Git.

Quick Start

Initial Setup:

```
cd /home/ubuntu/Sports-Bar-TV-Controller
./scripts/init-local-config.sh
# Edit for your system
nano config/local.local.json
```

After GitHub Updates:

```
git pull origin main
# Your local files remain untouched!
# If new options were added:
./scripts/init-local-config.sh  # Only adds missing options
```

Configuration Structure

`local.local.json`:

```
{
  "system": {
    "name": "Your Bar Name",
    "location": "City, State",
    "timezone": "America/New_York"
  },
  "network": {
    "apiPort": 3000,
    "devPort": 3001,
    "allowedOrigins": ["http://localhost:3000"]
  },
  "wolfpack": {
    "ip": "192.168.1.100",
    "port": 4999,
    "protocol": "tcp",
    "enabled": true
  },
  "features": {
    "aiAnalysis": true,
    "autoScheduling": true,
    "sportsFinder": true
  }
}
```

File Priority

1. **Template** (`*.template.json`) - Default values
2. **Local** (`*.local.json`) - Your overrides
3. **Environment** (`.env`) - Secrets and sensitive data
4. **Database** - Runtime configurations from UI

Later sources override earlier ones.

Configuration File Naming

Configuration files are automatically named based on your matrix configuration name for easier identification.

Automatic Config File Naming:
- Configuration files are automatically named based on your matrix configuration name
- Example: If your matrix is named "Wolfpack Controller", the config file becomes `wolfpack-controller.local.json`
- Automatic renaming happens during system updates

Benefits:
- Easy identification in multi-system setups
- Prevents confusion when managing multiple locations
- Backup files clearly show which system they belong to

Naming Convention:

```
Matrix Name: "Wolfpack Controller"
Config File: wolfpack-controller.local.json

Matrix Name: "Main Bar System"
Config File: main-bar-system.local.json
```

Backup & Restore System

The Sports Bar TV Controller includes a comprehensive backup and restore system with automatic and manual backup capabilities.

Automatic Backups

Backup on System Update:
- Every time you run `./update_from_github.sh`, a backup is automatically created
- Backup is created BEFORE pulling any changes from GitHub
- Ensures your configuration is safe even if something goes wrong

What Gets Backed Up:
- Configuration files (`config/*.local.json`)
- Environment variables (`.env`)
- Database with all your data (`prisma/dev.db` or PostgreSQL dump)
- Matrix configurations
- Device settings (DirecTV, FireTV, Cable boxes)
- Input/output mappings and scenes
- Audio zones and settings
- Sports guide configuration
- AI API keys
- Soundtrack credentials
- Data files (`data/*.json`)
- Streaming service credentials
- Device subscriptions
- Scene logs
- Atlas audio configs
- User uploads

Backup Retention:
- Keeps the last 10 backups automatically
- Older backups are automatically deleted
- Backups are stored in `~/sports-bar-backups/`

Backup Schedule:
- Daily execution at 3:00 AM (server time) via cron job
- Manual backups can be created anytime

Backup Script Location:
- `/home/ubuntu/Sports-Bar-TV-Controller/backup_script.js`

Backup Directory:
- `/home/ubuntu/Sports-Bar-TV-Controller/backups/`

Cron Job Configuration:

```
# Daily backup at 3:00 AM
0 3 * * * cd /home/ubuntu/Sports-Bar-TV-Controller && /usr/bin/node backup_script.js >
> /home/ubuntu/Sports-Bar-TV-Controller/backup.log 2>&1
```

Manual Backup & Restore Interface

Access: Navigate to `/backup-restore` from the main menu

Features:
- **Create Backup:** One-click backup of current configuration
- **View Recent Backups:** See the last 6 backups with:
- Timestamp
- File size
- Filename
- **Restore Backup:** Restore any previous backup
- Safety backup created automatically before restore
- Preserves current state before applying old backup
- **Delete Backup:** Remove old backups you no longer need

Web UI Location:
- `/backup-restore` - Backup & Restore page
- Accessible from main navigation in Configuration section

API Endpoints:
- `GET /api/backup` - Lists all available backups
- `POST /api/backup` - Create, restore, or delete backups

Using the Backup System

Create a Manual Backup

1. Go to Backup & Restore page
2. Click **Create Backup**
3. Wait for confirmation
4. New backup appears in the list

Restore from a Backup

1. Go to Backup & Restore page
2. Find the backup you want to restore
3. Click **Restore** button
4. Confirm the restoration
5. A safety backup is created automatically
6. Your system is restored to that backup state
7. **Restart the application** for changes to take effect:
   `bash`
   ```
   cd /home/ubuntu/Sports-Bar-TV-Controller
   pm2 restart sports-bar-tv-controller
   ```

Delete a Backup

1. Go to Backup & Restore page
2. Find the backup you want to delete
3. Click **Delete** button
4. Confirm deletion
5. Backup is permanently removed

Backup File Naming

Format: `backup_YYYY-MM-DD_HH-MM-SS.json` or `config-backup-YYYYMMDD-HHMMSS.tar.gz`
Example: `backup_2025-10-09_03-00-00.json`

Manual Backup (Command Line)

Create Manual Backup:

```
cd /home/ubuntu/Sports-Bar-TV-Controller
node backup_script.js
# OR
tar -czf ~/sports-bar-backups/manual-backup-$(date +%Y%m%d-%H%M%S).tar.gz \
  config/*.local.json .env prisma/dev.db data/*.json
```

Restore Manual Backup:

```
cd /home/ubuntu/Sports-Bar-TV-Controller
tar -xzf ~/sports-bar-backups/backup-filename.tar.gz
pm2 restart sports-bar-tv-controller
```

Backup Retention

Retention Policy:
- 14 days (backups older than 14 days are automatically deleted)
- Keeps last 10 backups automatically
- User can manually delete anytime

Verification:

```
# View recent backups
ls -lh ~/Sports-Bar-TV-Controller/backups/

# View backup log
tail -50 ~/Sports-Bar-TV-Controller/backup.log

# Verify cron job
crontab -l | grep backup
```

Safety Features

1. **Pre-Update Backup:** Automatic backup before GitHub pull
2. **Pre-Restore Safety Backup:** Automatic backup before restoring
3. **Backup Verification:** System verifies backup files exist before operations
4. **Atomic Operations:** Restore operations are atomic (all-or-nothing)
5. **Automatic Cleanup:** Automatic cleanup of old backups (keeps last 10)
6. **Error Handling:** Graceful error handling with clear messages

AI Features

AI Diagnostics System

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 providing real-time system health monitoring:

- **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
- Database Connectivity
- Matrix Inputs
- IR Devices
- Knowledge Base
- System Logs
- API Health
- **Quick Actions**: Links to AI Hub, System Admin, and Device Config
- **Real-time Updates**: Refresh button to re-run diagnostics on demand

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

Available diagnostic checks:

1. **Database Connectivity** - Tests database connection
2. **AI Providers** - Counts active vs total AI providers
3. **Matrix Inputs** - Checks Wolf Pack matrix input configuration
4. **IR Devices** - Validates IR device configuration file
5. **Device Mapping** - Analyzes matrix input to IR device mapping
6. **Knowledge Base** - Checks AI knowledge base directory
7. **System Logs** - Analyzes recent system logs
8. **API Health** - Tests critical API endpoints

API Usage:

```
// POST Request
{
  "checks": ["all"],  // or specific checks
  "detailed": false
}

// Response
{
  "success": true,
  "overallHealth": "healthy",
  "healthScore": 87,
  "summary": {
    "total": 8,
    "passed": 7,
    "failed": 0,
    "warnings": 1
  },
  "diagnostics": [ ... ]
}
```

3. Intelligent Troubleshooter Component

AI-powered device diagnostics and automated repair system:

- 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

How the AI Can Use Diagnostics

Proactive Monitoring:

```
// 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 })
})
```

Targeted Troubleshooting:

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

Quick Health Checks:

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

AI Knowledge System

The Sports Bar AI Assistant includes a powerful knowledge base system that has learned from **all system documentation**.

Knowledge Base Statistics

- **Total Document Chunks**: 559
- **PDF Documents**: 60 (equipment manuals, configuration guides)
- **Markdown Files**: 65 (system documentation, implementation guides)
- **Total Content**: 1,471,423 characters of technical documentation

Documentation Included

Equipment Manuals:
- Atlas IED Audio Processor (AZM4/AZM8) Documentation
- Wolf Pack HDMI Matrix Switcher
- Global Cache iTach IR Control
- DirecTV Control API
- Fire TV Integration

System Configuration:
- Matrix control configuration and setup
- Audio zone management
- TV input/output mapping
- Device discovery and CEC control
- Network configuration

Implementation Guides:
- AI enhancements and device insights
- Sports guide integration
- NFHS streaming setup
- Bartender remote interface
- Smart scheduler system
- Soundtrack integration
- Backup and restore procedures

Development Documentation:
- API reference
- Database schema
- Component architecture
- Styling standards
- GitHub synchronization
- Update procedures

How It Works

1. **Document Processing**: Processes all PDFs and Markdown files, extracts text, chunks large documents
2. **Knowledge Search**: Searches for relevant document chunks, ranks by relevance

3. **AI Response Generation**: AI receives question + relevant documentation context + source citations

Using the AI Assistant

Access: Navigate to `/ai-assistant` or click "AI Assistant" from the main menu

Features:
- **Knowledge Base Toggle**: Enable/disable documentation context
- **Contextual Answers**: Based on actual system documentation
- **Source Citations**: Shows which documents were referenced
- **Technical Accuracy**: Uses official manuals and guides
- **Real-time**: Powered by local Ollama AI (no cloud dependency)

Question Examples:
- "How do I configure the Atlas audio processor for the main bar zone?"
- "What are the IR commands for DirecTV channel control?"
- "TV 8 is showing no signal, how do I troubleshoot?"
- "Explain the bartender remote interface layout"

Rebuilding the Knowledge Base

From the UI:
1. Go to AI Assistant page
2. Click the refresh icon next to Knowledge Base stats
3. Wait for rebuild to complete

From Command Line:

```
cd /home/ubuntu/Sports-Bar-TV-Controller
./build-knowledge-base.sh
```

File Locations:
- **Knowledge Base**: `/data/ai-knowledge-base.json`
- **Build Script**: `/scripts/build-knowledge-base.ts`
- **Search Library**: `/src/lib/ai-knowledge.ts`

Q&A Training System

A comprehensive feature that allows you to train the AI assistant with domain-specific knowledge about your system.

Features

1. Automatic Q&A Generation

The system can automatically analyze your codebase and documentation to generate relevant question-answer pairs.

Supported Sources:
- **Repository**: Generates Q&As from README, INSTALLATION.md, and key documentation files
- **Documentation**: Analyzes all Markdown files in the docs/ folder
- **Codebase**: Examines TypeScript/JavaScript files to create Q&As about APIs

How it works:
1. Click "Generate from Repository" or "Generate from Docs" button

2. System creates a background job to process files
3. For each file, uses local AI model (Ollama) to generate 3-5 relevant Q&A pairs
4. Generated Q&As are automatically categorized and stored in database
5. Progress shown in real-time

Categories:
- `system`: System architecture and design
- `api`: API endpoints and usage
- `features`: System features and capabilities
- `configuration`: Setup and configuration
- `troubleshooting`: Common issues and solutions
- `general`: General information

2. Q&A Document Upload

Upload Q&A documents in multiple formats to quickly populate the training database.

Supported Formats:
- Q:/A: Format
- Question:/Answer: Format
- JSON Format
- Markdown Format

3. Q&A Management

View, filter, edit, and delete Q&A entries through the management interface.

Features:
- Filtering by category and source type
- Search across questions, answers, and tags
- Edit questions, answers, and categories
- Delete outdated or incorrect Q&As
- View usage statistics

API Endpoints:
- `/api/ai/qa-generate` - Generate Q&As
- `/api/ai/qa-upload` - Upload Q&A file
- `/api/ai/qa-entries` - List/manage Q&A entries

System Optimization

Caching System

A comprehensive in-memory caching service with:
- ✅ TTL (Time-To-Live) support
- ✅ Automatic cleanup of expired entries
- ✅ Pattern-based cache clearing
- ✅ Cache statistics and memory monitoring
- ✅ Get-or-set pattern for easy integration

Cache TTL Presets:
- SHORT: 1 minute

- MEDIUM: 5 minutes
- LONG: 15 minutes
- HOUR: 1 hour
- DAY: 24 hours

Cache Keys for:
- Sports guide data
- TV guide data
- Device status
- Device subscriptions
- Atlas analysis
- Soundtrack data
- NFHS streams
- AI analysis results

Implementation Location: `src/lib/cache-service.ts`

Usage Example:

```
import { cacheService, CacheKeys, CacheTTL } from '@/lib/cache-service'

// Simple get/set
cacheService.set('my-key', data, CacheTTL.MEDIUM)
const cached = cacheService.get('my-key')

// Get-or-set pattern
const data = await cacheService.getOrSet(
  CacheKeys.sportsGuide('nfl', '2025-10-01'),
  async () => {
    return await fetchNFLGames()
  },
  CacheTTL.HOUR
)
```

Performance Improvements

Before Optimization:
- ❌ Every request hit external APIs
- ❌ No caching = slow response times
- ❌ Mock data returned when APIs failed

After Optimization:
- ✅ Intelligent caching reduces API calls by ~80%
- ✅ Cache hits return data in <5ms vs 500-2000ms
- ✅ Real data or empty responses (no fake data)
- ✅ Cache statistics available for monitoring
- ✅ Automatic cleanup prevents memory leaks

Performance Gains:
- Sports guide data: 2000ms → 5ms (cached)
- Device subscriptions: 1500ms → 5ms (cached)
- Channel guide: 1000ms → 5ms (cached)

Real Device Subscription Detection:
Replaces ALL mock subscription data with real device polling:

- **DirecTV**: Connects to real receiver HTTP API, polls actual subscribed packages
- **Fire TV**: Connects via ADB, scans installed packages, detects streaming apps
- Cached for 1 hour to reduce API calls

Implementation Location: `src/lib/real-device-subscriptions.ts`

Mock Data Removal

Complete removal of mock/sample data throughout the application:

Removed From:
1. Device Subscriptions API (`src/app/api/device-subscriptions/poll/route.ts`)
2. Channel Guide API (`src/app/api/channel-guide/route.ts`)

New Behavior:
- Returns only real API data or empty with helpful messages
- Clear configuration instructions when APIs unavailable
- Proper error messages for debugging

Recent Changes and Fixes

October 10, 2025 - Atlas Zone Labels, Matrix Label Updates, and Matrix Test Fixes

1. Atlas Zone Output Labels Fixed

Issue: Zone labels in Audio Control Center were showing hardcoded “Matrix 1-4” instead of actual video input names.

Solution:
- Modified AudioZoneControl.tsx to fetch Matrix output labels from video-input-selection API
- Labels now dynamically reflect selected video input names (e.g., “Cable Box 1” instead of “Matrix 1”)
- Component refreshes automatically when video input selection changes

Files Modified:
- `src/components/AudioZoneControl.tsx`

2. Matrix Label Dynamic Updates Implemented

Issue: Matrix labels weren’t updating dynamically when video inputs were selected.

Solution:
- Added cross-component communication mechanism using window object
- AudioZoneControl exposes `refreshConfiguration()` function
- MatrixControl calls this function after successful video input selection
- Labels update immediately in both Audio Control Center and Bartender Remote

Files Modified:
- `src/components/AudioZoneControl.tsx` - Added refresh mechanism
- `src/components/MatrixControl.tsx` - Added refresh trigger

3. Matrix Test Database Error Fixed

Issue: Wolf Pack Connection Test was failing with database error.

Root Cause: testLog.create() calls were not properly handling nullable fields.

Solution:
- Updated test routes to ensure proper data types for all fields
- Added explicit null values for optional fields
- Improved error handling with try-catch blocks

Files Modified:
- `src/app/api/tests/wolfpack/connection/route.ts`
- `src/app/api/tests/wolfpack/switching/route.ts`

October 10, 2025 - CRITICAL: Atlas Configuration Restoration and Bug Fixes

Critical Bug Fixed: Configuration Wipe

Problem: The upload/download configuration feature had a critical bug generating random configuration data instead of reading from the actual Atlas processor.

Solution Implemented:
1. `src/app/api/atlas/download-config/route.ts` - Fixed to read from saved configuration file
2. `src/app/api/atlas/upload-config/route.ts` - Saves configuration to file system BEFORE attempting processor upload

Atlas Configuration Restored

Processor Information:
- **Model**: AZMP8 (8 inputs, 8 outputs, 8 zones)
- **IP Address**: 192.168.5.101:80
- **Processor ID**: cmgjxa5ai0000260a7xuiepjl
- **Name**: Graystone Alehouse Main Audio
- **Status**: Online and authenticated

Configuration Backup Location:
- **Primary Config**: `/home/ubuntu/Sports-Bar-TV-Controller/data/atlas-configs/cmgjx-a5ai0000260a7xuiepjl.json`
- **Backups**: `/home/ubuntu/Sports-Bar-TV-Controller/data/atlas-configs/cmgjx-a5ai0000260a7xuiepjl_backup_*.json`

Restored Configuration:
- 7 Inputs Configured (Matrix 1-4, Mic 1-2, Spotify)
- 7 Outputs Configured (Bar, Bar Sub, Dining Room, Party Room West/East, Patio, Bathroom)
- 3 Scenes Configured

October 10, 2025 - Sports Guide API Integration and Atlas AI Monitor Fix

Sports Guide API Integration

API Provider: The Rail Media
API Endpoint: https://guide.thedailyrail.com/api/v1
User ID: 258351

Implementation Details:
- API Service Client (`src/lib/sportsGuideApi.ts`)
- API Routes for status, verification, channel data
- UI Component (`src/components/SportsGuideConfig.tsx`)
- Security: API key stored in `.env` file, masked in UI

Atlas AI Monitor Fix

Issue: The Atlas AI Monitor component was using hardcoded processor values instead of actual database data.

Solution: Updated Audio Control Center page to fetch active processor data dynamically.

Files Modified:
- `src/app/audio-control/page.tsx`

October 9, 2025 - Outputs 1-4 Configuration Update

Issue

Outputs 1-4 were configured as "simple outputs" with limited controls, inconsistent with actual hardware setup where outputs 1-4 are full matrix outputs.

Solution

Code Changes:
- Modified `src/components/MatrixControl.tsx`
- Changed `isSimpleOutput` flag from `true` to `false` for outputs 1-4
- Added full controls: power on/off, active checkbox, audio output field

Result:
Outputs 1-4 now display full controls matching their actual matrix configuration.

October 9, 2025 - Automated Backup System

Implemented automated daily backup system for matrix configuration and database files.

Schedule: Daily execution at 3:00 AM (server time)
Retention: 14 days
Status: ✅ Verified and operational

Additional Changes (October 2025)

Wolfpack Connection Test Fix

Issue: Connection test consistently failing with database-related errors.

Root Cause: Duplicate PrismaClient instantiation

Solution: Used Prisma singleton from `@/lib/prisma`

Files Modified: `src/pages/api/wolfpack/test-connection.ts`

TV Selection Options Restoration

Issue: Missing UI controls for TV selection in schedules.

Solution: Added `dailyTurnOn` and `dailyTurnOff` boolean fields to MatrixOutput schema

EPG Services Removal

Issue: Application contained references to deprecated/unavailable EPG services.

Services Removed:
- Gracenote API
- TMS (Tribune Media Services)
- Spectrum Business API

Files Modified:
- `src/components/ApiKeysManager.tsx`
- `src/components/SportsGuide.tsx`
- `src/components/tv-guide/TVGuideConfigurationPanel.tsx`

Troubleshooting

Wolfpack Connection Test Fails

Symptoms: Connection test returns error, “Failed to connect” message

Solutions:

1. **Verify Network Connectivity**:
   ```
   bash
      ping <wolfpack-ip-address>
      curl http://<wolfpack-ip-address>
   ```

2. **Check Wolfpack Matrix**:
   - Ensure matrix is powered on
   - Verify IP address is correct
   - Check network cable connections
   - Confirm matrix is on same network/VLAN

3. **Verify Database Connection**:
   ```
   bash
      npx prisma db pull
   ```

4. **Check Application Logs**:
   ```
   bash
      pm2 logs sports-bar-tv-controller
   ```

5. **Restart Application**:
   ```
   bash
      pm2 restart sports-bar-tv-controller
   ```

Matrix Switching Not Working

Symptoms: Routing commands fail, TVs don’t change sources

Solutions:

1. **Test Connection First**: Use connection test in System Admin
2. **Check Output Configuration**: Ensure outputs are enabled
3. **Verify Input Configuration**: Ensure inputs are enabled

4. **Test Individual Commands**: Try routing single input to single output

TV Selection Not Working

Symptoms: All TVs turn on despite dailyTurnOn settings

Solutions:

1. **Verify Database Migration**:
   ```bash
      npx prisma migrate status
      npx prisma migrate deploy
   ```

2. **Check Output Configuration**: Navigate to System Admin → Matrix Outputs

3. **Restart Application**:
   ```bash
      pm2 restart sports-bar-tv-controller
   ```

Configuration Lost After Update

Symptoms: Wolfpack IP address missing, input/output labels reset

Solutions:

1. **Check Database**:
   ```bash
      npx prisma studio
   ```

2. **Restore from Backup**:
   ```bash
      # Navigate to /backup-restore page
      # Or use command line:
      cd ~/Sports-Bar-TV-Controller
      tar -xzf ~/sports-bar-backups/latest-backup.tar.gz
      pm2 restart sports-bar-tv-controller
   ```

3. **Reconfigure Manually**: Re-enter settings through UI

Atlas Shows Offline

Solutions:
1. Check network connectivity: `ping 192.168.5.101`
2. Check port accessibility: `nc -zv 192.168.5.101 80`
3. Verify Atlas processor is powered on
4. Check firewall rules

API Routes Not Working

Problem: Getting 404 errors on API routes

Solution:
1. Check if `pages/api` directory exists - if yes, delete it
2. Use production build: `npm run build && npm start`
3. Clear Next.js cache: `rm -rf .next`

Backup Failed to Create

Solutions:
- Check disk space: `df -h`
- Check permissions: `ls -la ~/sports-bar-backups/`
- Check error message in the UI
- Review backup logs: `tail -50 ~/Sports-Bar-TV-Controller/backup.log`

AI Assistant Not Responding

Solutions:
1. Verify Ollama is running: `systemctl status ollama`
2. Check Ollama URL: `http://localhost:11434`
3. Review AI configuration in `/ai-config`
4. Check system resources: `htop`

Deployment Guide

Prerequisites

- Node.js 18+ installed
- PostgreSQL database running (or SQLite for development)
- PM2 process manager installed
- Git repository access
- SSH access to server (port 224)

Initial Deployment

1. **Clone Repository**:
   ```
   bash
      cd ~
      git clone https://github.com/dfultonthebar/Sports-Bar-TV-Controller.git
      cd Sports-Bar-TV-Controller
   ```

2. **Install Dependencies**:
   ```
   bash
      npm install
   ```

3. **Configure Environment**:
   ```
   bash
      cp .env.example .env
      nano .env
   ```
   Set:
   - `DATABASE_URL`: PostgreSQL connection string
   - `NEXTAUTH_SECRET`: Random secret for authentication
   - `NEXTAUTH_URL`: Application URL
   - `OLLAMA_BASE_URL`: Ollama API URL (default: http://localhost:11434)
   - Sports Guide API credentials
   - Other API keys as needed

4. **Initialize Configuration**:
   `bash`
   ```
      ./scripts/init-local-config.sh
   ```

5. **Initialize Database**:
   `bash`
   ```
      npx prisma migrate deploy
      npx prisma generate
   ```

6. **Build Application**:
   `bash`
   ```
      npm run build
   ```

7. **Start with PM2**:
   `bash`
   ```
      pm2 start npm --name "sports-bar-tv-controller" -- start
      pm2 save
      pm2 startup  # Follow instructions to enable auto-start
   ```

Update Deployment

1. **SSH into Server**:
   `bash`
   ```
      ssh -p 224 ubuntu@24.123.87.42
   ```

2. **Navigate to Project**:
   `bash`
   ```
      cd ~/Sports-Bar-TV-Controller
   ```

3. **Run Update Script**:
   `bash`
   ```
      ./update_from_github.sh
   ```
   This script automatically:
   - Creates a backup
   - Pulls latest changes
   - Installs dependencies
   - Runs database migrations
   - Renames config files based on matrix name
   - Builds application
   - Restarts PM2

OR Manual Update:

1. **Pull Latest Changes**:
   `bash`
   ```
      git pull origin main
   ```

2. **Install Dependencies**:
   `bash`
   ```
      npm install
   ```

3. **Run Migrations**:
   `bash`

```
   npx prisma migrate deploy
   npx prisma generate
```

4. **Rebuild Application**:
   ```bash
   npm run build
   ```

5. **Restart PM2**:
   ```bash
   pm2 restart sports-bar-tv-controller --update-env
   ```

Rollback Procedure

If deployment fails:

1. **Revert Git Changes**:
   ```bash
   git log  # Find previous commit hash
   git checkout <previous-commit-hash>
   ```

2. **Rebuild and Restart**:
   ```bash
   npm install
   npm run build
   pm2 restart sports-bar-tv-controller
   ```

3. **Rollback Database** (if needed):
   ```bash
   # Restore from backup
   cd /home/ubuntu/Sports-Bar-TV-Controller
   # Use backup/restore UI or command line
   tar -xzf ~/sports-bar-backups/pre-update-backup.tar.gz
   pm2 restart sports-bar-tv-controller
   ```

Maintenance and Updates

Regular Maintenance Tasks

Daily

- Monitor PM2 logs for errors:
  ```bash
  pm2 logs sports-bar-tv-controller --lines 100
  ```
- Check automatic backup execution
- Review system health score in AI Diagnostics

Weekly

- Check disk space: `df -h`
- Review application logs for warnings
- Verify scheduled tasks are running
- Test backup restoration
- Review AI diagnostics reports

Monthly

- Update dependencies:
  `bash`
  `  npm update`
  `  npm audit fix`
- Review and optimize database
- Clean old log files
- Review backup retention
- Update system documentation
- Review AI knowledge base

Backup Strategy

Automated Daily Backup:
- Runs at 3:00 AM via cron job
- Backs up config files, database, data files
- Retention: 14 days
- Location: `~/Sports-Bar-TV-Controller/backups/`

Manual Backup:

```
# Via UI: /backup-restore page
# OR via command line:
cd ~/Sports-Bar-TV-Controller
node backup_script.js
```

Backup Retention:
- Daily backups: Keep 7 days
- Weekly backups: Keep 4 weeks
- Monthly backups: Keep 12 months
- Critical backups: Keep indefinitely (download to external storage)

Restore from Backup:

```
# Via UI: /backup-restore page
# OR via command line:
cd ~/Sports-Bar-TV-Controller
tar -xzf ~/sports-bar-backups/backup-YYYYMMDD-HHMMSS.tar.gz
pm2 restart sports-bar-tv-controller
```

Monitoring

Application Health

```
# Check PM2 status
pm2 status

# View resource usage
pm2 monit

# Check application logs
pm2 logs sports-bar-tv-controller

# View error logs only
pm2 logs sports-bar-tv-controller --err
```

Database Health

```
# Check database size
psql -d sports_bar_tv -c "SELECT pg_size_pretty(pg_database_size('sports_bar_tv'));"

# Check table sizes
psql -d sports_bar_tv -c "SELECT schemaname, tablename,
pg_size_pretty(pg_total_relation_size(schemaname||'.'||tablename)) FROM pg_tables
WHERE schemaname = 'public' ORDER BY pg_total_relation_size(schemaname||'.'||
tablename) DESC;"

# For SQLite (development):
ls -lh prisma/dev.db
```

Network Connectivity

```
# Test Wolfpack connectivity
curl http://<wolfpack-ip-address>

# Test Atlas connectivity
curl http://192.168.5.101

# Check application port
netstat -tulpn | grep 3001
```

System Resources

```
# Check disk space
df -h

# Check memory usage
free -h

# Check CPU usage
top

# Check process details
ps aux | grep node
```

Updating Dependencies

Check for Updates:

```
npm outdated
```

Update All Dependencies:

```
npm update
```

Update Specific Package:

```
npm update <package-name>
```

Security Audit:

```
npm audit
npm audit fix
```

Rebuilding AI Knowledge Base

When to Rebuild:
- After uploading new equipment manuals
- When documentation is updated
- After system configuration changes
- New features are documented

How to Rebuild:

```
# Via UI: /ai-assistant page (click refresh icon)
# OR via command line:
cd ~/Sports-Bar-TV-Controller
./build-knowledge-base.sh
```

Security Considerations

Network Security

- Wolfpack matrix should be on isolated VLAN
- Atlas processor should be on secure network
- Application should be behind firewall
- Use HTTPS in production (configure reverse proxy)
- SSH configured on non-standard port 224

Authentication

- Change default admin credentials immediately
- Use strong passwords (SSH password already complex)
- Enable two-factor authentication if available

- Regular password rotation policy
- Limit SSH access to authorized IPs if possible

Database Security

- Use strong database passwords
- Restrict database access to localhost
- Regular security updates
- Encrypted backups for sensitive data
- Regular audit of database access logs

Application Security

- Keep dependencies updated: `npm audit fix`
- Regular security audits
- Monitor logs for suspicious activity
- API rate limiting enabled
- Input validation on all endpoints
- CORS properly configured

API Key Management

- Store API keys in `.env` file (gitignored)
- Never commit API keys to repository
- Rotate API keys periodically
- Use environment-specific keys
- Monitor API key usage

File Upload Security

- Validate file types
- Scan uploaded files for malware
- Limit file sizes
- Store uploads outside web root
- Regular cleanup of old uploads

Backup Security

- Encrypt backups containing sensitive data
- Store backups in secure location
- Test backup restoration regularly
- Implement backup versioning
- Off-site backup storage for critical data

Support and Resources

Documentation

- **This Document**: Complete system documentation
- **Next.js**: https://nextjs.org/docs
- **Prisma**: https://www.prisma.io/docs

- **Wolfpack API**: Refer to Wolfpack matrix documentation
- **Atlas IED**: Refer to Atlas processor manuals (in uploads/)

Common Issues

- See Troubleshooting section
- Check GitHub Issues: https://github.com/dfultonthebar/Sports-Bar-TV-Controller/issues

Getting Help

1. **Check Documentation**: Review this document and specific feature docs
2. **AI Assistant**: Ask the AI assistant (`/ai-assistant`)
3. **AI Diagnostics**: Run system diagnostics (`/ai-diagnostics`)
4. **Application Logs**: Check PM2 logs for errors
5. **GitHub Issues**: Search existing issues or create new one

When Creating GitHub Issue:
- Detailed description of the problem
- Steps to reproduce
- Error messages and logs
- System information (OS, Node version, etc.)
- Screenshots if applicable

Best Practices

System Management:
1. Regular backups before making changes
2. Test in development before production
3. Monitor system health regularly
4. Keep documentation up to date
5. Use version control for custom changes

Device Configuration:
1. Label devices clearly and consistently
2. Document custom configurations
3. Test changes before applying to all devices
4. Keep firmware updated on hardware devices
5. Maintain device inventory

AI System:
1. Rebuild knowledge base after documentation updates
2. Review and edit auto-generated Q&As
3. Add custom Q&As for site-specific knowledge
4. Monitor AI diagnostics regularly
5. Keep Ollama models updated

Changelog

October 2025

- ✅ Fixed Wolfpack connection test (PrismaClient singleton)
- ✅ Added TV selection options (dailyTurnOn, dailyTurnOff)

- ✅ Removed EPG services (Gracenote, TMS, Spectrum Business API)
- ✅ Fixed outputs 1-4 configuration (full matrix controls)
- ✅ Implemented automated backup system
- ✅ Fixed Atlas zone labels (dynamic video input names)
- ✅ Implemented matrix label dynamic updates
- ✅ Fixed matrix test database errors
- ✅ Restored Atlas configuration from backup
- ✅ Integrated Sports Guide API (The Rail Media)
- ✅ Fixed Atlas AI Monitor component
- ✅ Updated system documentation
- ✅ Implemented local configuration system
- ✅ Implemented caching system for performance
- ✅ Removed all mock data (real device polling)
- ✅ Enhanced AI diagnostics system
- ✅ Built comprehensive AI knowledge base
- ✅ Implemented Q&A training system

September 2025

- Initial system deployment
- Basic matrix control implementation
- Sports guide integration
- Bartender remote interface

Future Enhancements

- [ ] Custom EPG service integration
- [ ] Enhanced backup automation with cloud storage
- [ ] Mobile app development
- [ ] Advanced scheduling features
- [ ] Multi-location management
- [ ] Voice control integration
- [ ] Automated device discovery
- [ ] Performance analytics dashboard
- [ ] Integration with additional audio processors
- [ ] Enhanced AI capabilities (predictive maintenance)

License

[Add license information here]

Contributors

[Add contributor information here]

Last Updated: October 10, 2025
Version: 2.0
Document Status: Complete - Single Source of Truth

This document is the comprehensive, authoritative source for all Sports Bar TV Controller system documentation. All other documentation files have been merged into this single document for consistency and ease of maintenance.