🚀 Sports Bar TV Controller - Comprehensive Deployment Guide

Complete guide for deploying the Sports Bar TV Controller in various environments.

📋 Table of Contents

🎯 Quick Start

One-Line Installation

The fastest way to get started:

```
curl -sSL https://raw.githubusercontent.com/dfultonthebar/Sports-Bar-TV-Controller/
main/install.sh | bash
```

Installation Time: 5-10 minutes (depending on internet speed)

What Happens:
1. System checks and prerequisites validation
2. Node.js v22 installation
3. Ollama AI platform setup
4. AI model downloads (4 models, ~15GB)
5. Application cloning and dependency installation
6. Database creation and migration
7. Knowledge base building (2,700+ chunks)
8. Application build and deployment
9. Optional systemd service configuration

Result: Fully functional Sports Bar TV Controller at http://localhost:3000

💻 System Requirements

Minimum Requirements

Component	Minimum	Recommended
OS	Ubuntu 20.04+ / Debian 11+ (64-bit)	Ubuntu 22.04 LTS
CPU	2 cores	4+ cores (Intel i5 or better)
RAM	4GB	16GB+
Disk	10GB free	50GB+ SSD
Network	10 Mbps	Gigabit Ethernet

Recommended Production Hardware

Intel NUC13ANHi5 or equivalent:
- **CPU:** Intel Core i5-1340P (12 cores, 16 threads)
- **RAM:** 16GB DDR4 (32GB for heavy AI usage)
- **Storage:** 512GB NVMe SSD
- **Network:** Gigabit Ethernet
- **USB:** Multiple USB 3.0 ports for CEC adapter

Software Prerequisites

Automatically installed by the installer:
- Node.js v22.x
- npm (comes with Node.js)
- SQLite (embedded, no separate installation)
- Ollama AI platform
- libCEC drivers (for HDMI-CEC control)

Optional (for advanced features):
- Docker (for containerized deployment)
- Nginx (for reverse proxy)
- Let’s Encrypt (for HTTPS)

Network Requirements

- **Outbound HTTPS (443):** For downloading packages and AI models
- **Inbound HTTP (3000):** For web interface access
- **Local Network:** Access to matrix switchers, IR controllers, and TVs

Disk Space Breakdown

- **Application:** ~500MB
- **Node.js & Dependencies:** ~1GB
- **Ollama Platform:** ~500MB
- **AI Models:** ~15GB (4 models)
- **Database:** ~10MB (grows with usage)

- **Knowledge Base:** ~10MB
- **Logs:** ~100MB (with rotation)
- **Recommended Free Space:** 20GB+ for updates and growth

🛠️ Installation Methods

Method 1: Default Home Directory Installation (Recommended)

Best for: Single-user systems, development, testing

```
curl -sSL https://raw.githubusercontent.com/dfultonthebar/Sports-Bar-TV-Controller/
main/install.sh | bash
```

Installation Path: `$HOME/Sports-Bar-TV-Controller`

Benefits:
- No sudo required for most operations
- Simple permissions management
- Easy to update and maintain
- Runs as your current user
- Quick setup and teardown

Limitations:
- Not ideal for multi-user systems
- No automatic startup on boot (without sudo)
- User-specific installation

Method 2: Custom Directory Installation

Best for: Specific directory requirements, shared installations

```
curl -sSL https://raw.githubusercontent.com/dfultonthebar/Sports-Bar-TV-Controller/
main/install.sh | INSTALL_DIR=/custom/path bash
```

Examples:

```
# Install to /opt (system-wide)
curl -sSL https://raw.githubusercontent.com/dfultonthebar/Sports-Bar-TV-Controller/
main/install.sh | INSTALL_DIR=/opt/sportsbar bash

# Install to shared directory
curl -sSL https://raw.githubusercontent.com/dfultonthebar/Sports-Bar-TV-Controller/
main/install.sh | INSTALL_DIR=/srv/sportsbar bash

# Install to mounted volume
curl -sSL https://raw.githubusercontent.com/dfultonthebar/Sports-Bar-TV-Controller/
main/install.sh | INSTALL_DIR=/mnt/data/sportsbar bash
```

Note: System-wide installations (e.g., `/opt`, `/usr`) automatically create a service user.

Method 3: Skip Ollama Installation

Best for: Systems with Ollama already installed

```
curl -sSL https://raw.githubusercontent.com/dfultonthebar/Sports-Bar-TV-Controller/
main/install.sh | SKIP_OLLAMA=true bash
```

Requirements:
- Ollama must be installed and running
- Required models must be available: `llama3.2:latest`, `llama2:latest`, `mistral:latest`, `phi3:mini`

Verify Ollama:

```
ollama list
systemctl status ollama
```

Method 4: Specify Branch

Best for: Testing development branches, beta features

```
curl -sSL https://raw.githubusercontent.com/dfultonthebar/Sports-Bar-TV-Controller/
main/install.sh | REPO_BRANCH=develop bash
```

Available branches:
- `main` - Stable production release
- `develop` - Latest development features
- `feature/*` - Specific feature branches

Method 5: Combined Options

Combine multiple options:

```
# Custom directory + skip Ollama + specific branch
curl -sSL https://raw.githubusercontent.com/dfultonthebar/Sports-Bar-TV-Controller/
main/install.sh | \
  INSTALL_DIR=/opt/sportsbar \
  SKIP_OLLAMA=true \
  REPO_BRANCH=develop \
  bash
```

Method 6: Manual Installation

Best for: Advanced users, custom configurations, troubleshooting

```
# 1. Install Node.js v22
curl -fsSL https://deb.nodesource.com/setup_22.x | sudo -E bash -
sudo apt-get install -y nodejs

# 2. Install Ollama
curl -fsSL https://ollama.com/install.sh | sh

# 3. Pull AI models
ollama pull llama3.2:latest
ollama pull llama2:latest
ollama pull mistral:latest
ollama pull phi3:mini

# 4. Clone repository
git clone https://github.com/dfultonthebar/Sports-Bar-TV-Controller.git
cd Sports-Bar-TV-Controller

# 5. Install dependencies
npm install

# 6. Setup database
npx prisma generate
npx prisma db push

# 7. Build knowledge base
npm run build:kb

# 8. Build application
npm run build

# 9. Start application
npm start
```

✅ Post-Installation

1. Verify Installation

Check application status:

```
curl http://localhost:3000
```

Expected response: HTML content from the home page

Check AI system:

```
curl http://localhost:3000/api/ai/status
```

Expected response:

```
{
  "status": "operational",
  "ollama": "running",
  "models": ["llama3.2:latest", "llama2:latest", "mistral:latest", "phi3:mini"],
  "knowledgeBase": {
    "chunks": 2757,
    "size": "6.0 MB"
  }
}
```

Check Ollama models:

```
ollama list
```

Expected output:

```
NAME                    ID              SIZE      MODIFIED
llama3.2:latest         a80c4f17acd5    2.0 GB    2 minutes ago
llama2:latest           78e26419b446    3.8 GB    5 minutes ago
mistral:latest          f974a74358d6    4.1 GB    8 minutes ago
phi3:mini               4abea9e4f1e5    2.3 GB    10 minutes ago
```

2. Access the Application

Local access:

```
http://localhost:3000
```

Network access:

```
http://[your-server-ip]:3000
```

Find your server IP:

```
hostname -I | awk '{print $1}'
```

3. Test Core Features

Home page:
- Navigate to http://localhost:3000
- Verify dashboard loads
- Check for any error messages

AI Hub:
- Navigate to http://localhost:3000/ai-hub
- Try asking a question: “How do I control the matrix?”
- Verify AI responds with streaming text

Matrix Control:
- Navigate to http://localhost:3000/matrix

- Check if matrix controls are visible
- Test input/output selection (if hardware connected)

Device Config:
- Navigate to http://localhost:3000/device-config
- Verify device configuration interface loads
- Check TV, audio, and IR device sections

4. Configure Systemd Service (Optional)

If you have sudo access and want automatic startup:

The installer prompts for systemd service creation. If you skipped it, you can set it up manually:

```
# Create service file
sudo tee /etc/systemd/system/sportsbar-assistant.service > /dev/null <<EOF
[Unit]
Description=Sports Bar TV Controller
After=network.target ollama.service

[Service]
Type=simple
User=$USER
WorkingDirectory=$HOME/Sports-Bar-TV-Controller
ExecStart=/usr/bin/npm start
Restart=always
RestartSec=10
Environment=NODE_ENV=production

[Install]
WantedBy=multi-user.target
EOF

# Reload systemd
sudo systemctl daemon-reload

# Enable and start service
sudo systemctl enable sportsbar-assistant
sudo systemctl start sportsbar-assistant

# Check status
sudo systemctl status sportsbar-assistant
```

5. Configure Firewall (If Enabled)

```
# Allow HTTP traffic on port 3000
sudo ufw allow 3000/tcp

# Or allow from specific network only
sudo ufw allow from 192.168.1.0/24 to any port 3000

# Check firewall status
sudo ufw status
```

6. Test AI Features

Rebuild knowledge base:

```
cd ~/Sports-Bar-TV-Controller
npm run build:kb
```

Verify AI responses:

```
curl -X POST http://localhost:3000/api/ai/chat \
  -H "Content-Type: application/json" \
  -d '{"message": "How do I control the matrix?"}'
```

7. Run System Benchmark (Optional)

Establish performance baseline:

```
cd ~/Sports-Bar-TV-Controller
./scripts/system-benchmark.sh --quick
```

View results:

```
cat $(ls -t benchmark-reports/baseline-report-*.md | head -1)
```

🎯 Deployment Scenarios

Scenario 1: Fresh Installation on Intel NUC

Hardware: Intel NUC13ANHi5 with 16GB RAM, 512GB SSD

Steps:

1. **Install Ubuntu Server 22.04 LTS**
   ```
   bash
      # Download Ubuntu Server ISO
      # Create bootable USB
      # Install Ubuntu with default options
   ```

2. **Update system**
   ```
   bash
      sudo apt update && sudo apt upgrade -y
   ```

3. **Run installer**
   ```
   bash
      curl -sSL https://raw.githubusercontent.com/dfultonthebar/Sports-Bar-TV-Controller/main/
   install.sh | bash
   ```

4. **Configure network**
   ```
   bash
      # Set static IP (optional)
      sudo nano /etc/netplan/00-installer-config.yaml
   ```

5. **Enable systemd service**
   ```
   bash
   ```

```
   sudo systemctl enable sportsbar-assistant
   sudo systemctl start sportsbar-assistant
```

6. **Configure firewall**
   `bash`
   ```
      sudo ufw allow 3000/tcp
      sudo ufw enable
   ```

7. **Test from another device**
   `http://[nuc-ip-address]:3000`

See NUC_DEPLOYMENT.md (./NUC_DEPLOYMENT.md) for detailed NUC-specific instructions.

Scenario 2: Updating Existing Installation

Situation: You have an older version installed and want to update

Steps:

1. **Backup current installation**
   `bash`
   ```
      cd ~/Sports-Bar-TV-Controller
      cp prisma/data/sports_bar.db ~/sports_bar.db.backup-$(date +%Y%m%d)
   ```

2. **Run update script**
   `bash`
   ```
      ./update_from_github.sh
   ```

3. **Verify update**
   `bash`
   ```
      git log -1
      npm list | head -5
   ```

4. **Test application**
   `http://localhost:3000`

Note: Settings and database are automatically preserved during updates.

Scenario 3: Multiple Machine Deployment

Situation: Deploy to multiple sports bar locations

Approach 1: Manual deployment to each machine

```
# On each machine
curl -sSL https://raw.githubusercontent.com/dfultonthebar/Sports-Bar-TV-Controller/
main/install.sh | bash
```

Approach 2: Automated deployment with Ansible

Create `deploy.yml` :

```
---
- hosts: sportsbars
  become: yes
  tasks:
    - name: Run Sports Bar installer
      shell: |
        curl -sSL https://raw.githubusercontent.com/dfultonthebar/Sports-Bar-TV-
Controller/main/install.sh | bash
      args:
        creates: /home/ubuntu/Sports-Bar-TV-Controller
```

Run deployment:

```
ansible-playbook -i inventory.ini deploy.yml
```

Approach 3: Docker deployment

```
# Build Docker image
docker build -t sportsbar-controller .

# Deploy to multiple machines
docker run -d -p 3000:3000 --name sportsbar sportsbar-controller
```

Scenario 4: Production Deployment with Reverse Proxy

Situation: Deploy with HTTPS and custom domain

Steps:

1. **Install Nginx**
   `bash`
   `sudo apt install nginx -y`

2. **Configure Nginx**
   `bash`
   `sudo nano /etc/nginx/sites-available/sportsbar`

```nginx
server {
listen 80;
server_name sportsbar.example.com;

```
    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
```
```

~~~
}
```
~~~

1. **Enable site**
   ```bash
   sudo ln -s /etc/nginx/sites-available/sportsbar /etc/nginx/sites-enabled/
   sudo nginx -t
   sudo systemctl restart nginx
   ```

2. **Install SSL certificate**
   ```bash
   sudo apt install certbot python3-certbot-nginx -y
   sudo certbot --nginx -d sportsbar.example.com
   ```

3. **Test HTTPS access**
   `https://sportsbar.example.com`

Scenario 5: Development Environment

Situation: Set up for development and testing

Steps:

1. **Install to home directory**
   ```bash
   curl -sSL https://raw.githubusercontent.com/dfultonthebar/Sports-Bar-TV-Controller/main/install.sh | bash
   ```

2. **Start in development mode**
   ```bash
   cd ~/Sports-Bar-TV-Controller
   npm run dev
   ```

3. **Enable hot reload**
   - Edit files in `src/`
   - Changes automatically reload in browser

4. **Use Prisma Studio for database**
   ```bash
   npx prisma studio
   ```

5. **Test AI features**
   ```bash
   npm run check:ai
   ```

Scenario 6: Containerized Deployment

Situation: Deploy using Docker for isolation

Create Dockerfile:

```
FROM node:22-slim

WORKDIR /app

# Install dependencies
COPY package*.json ./
RUN npm install

# Copy application
COPY . .

# Build application
RUN npx prisma generate
RUN npm run build

# Expose port
EXPOSE 3000

# Start application
CMD ["npm", "start"]
```

Build and run:

```
docker build -t sportsbar-controller .
docker run -d -p 3000:3000 -v sportsbar-data:/app/prisma/data --name sportsbar sports-
bar-controller
```

Note: Ollama must be installed on the host or in a separate container.

⚙️ Configuration

Environment Variables

Create or edit `.env` file:

```
cd ~/Sports-Bar-TV-Controller
nano .env
```

Common variables:

```
# Application
NODE_ENV=production
PORT=3000

# Database
DATABASE_URL="file:./data/sports_bar.db"

# Ollama
OLLAMA_BASE_URL=http://localhost:11434
OLLAMA_MODEL=llama3.2:latest

# AI Features
AI_TIMEOUT=180000
AI_MAX_TOKENS=4096
AI_TEMPERATURE=0.7

# Security
SESSION_SECRET=your-secret-key-here
ALLOWED_ORIGINS=http://localhost:3000,http://192.168.1.100:3000

# Logging
LOG_LEVEL=info
LOG_FILE=logs/app.log
```

Database Configuration

Default location: `prisma/data/sports_bar.db`

Change database location:

```
DATABASE_URL="file:/custom/path/sports_bar.db"
```

Database operations:

```
# Generate Prisma client
npx prisma generate

# Push schema changes
npx prisma db push

# Reset database (WARNING: deletes all data)
npx prisma db push --force-reset

# Open Prisma Studio
npx prisma studio
```

Ollama Configuration

Change Ollama URL:

```
OLLAMA_BASE_URL=http://different-host:11434
```

Change default model:

```
OLLAMA_MODEL=mistral:latest
```

Verify Ollama connection:

```
curl http://localhost:11434/api/tags
```

AI Configuration

Adjust AI timeouts:

```
AI_TIMEOUT=180000  # 3 minutes in milliseconds
```

Change AI model parameters:

```
AI_TEMPERATURE=0.7  # 0.0 = deterministic, 1.0 = creative
AI_MAX_TOKENS=4096  # Maximum response length
```

Rebuild knowledge base:

```
npm run build:kb
```

Network Configuration

Change application port:

```
PORT=8080
```

Bind to specific interface:

```
HOST=0.0.0.0  # Listen on all interfaces
# or
HOST=192.168.1.100  # Listen on specific IP
```

Configure CORS:

```
ALLOWED_ORIGINS=http://localhost:3000,http://192.168.1.100:3000,https://sports-
bar.example.com
```

Hardware Configuration

Matrix switcher:
- Configure in Device Config page
- Set IP address and port
- Test connection

IR controllers:
- Configure Global Cache devices
- Set IP addresses
- Test IR commands

CEC adapter:
- Plug in Pulse-Eight USB adapter

- Run CEC discovery
- Configure TV power control

🔧 Troubleshooting

Installation Issues

Issue: Installation fails immediately

Symptoms:

```
curl: (7) Failed to connect to raw.githubusercontent.com port 443
```

Solutions:
1. Check internet connection
2. Verify DNS resolution: `nslookup raw.githubusercontent.com`
3. Try alternative download method:
`bash`
```
  wget https://raw.githubusercontent.com/dfultonthebar/Sports-Bar-TV-Controller/main/install.sh
  bash install.sh
```

Issue: Node.js installation fails

Symptoms:

```
E: Unable to locate package nodejs
```

Solutions:
1. Manually add NodeSource repository:
`bash`
```
  curl -fsSL https://deb.nodesource.com/setup_22.x | sudo -E bash -
  sudo apt-get install -y nodejs
```

1. Verify Node.js version:
   `bash`
   ```
     node --version  # Should show v22.x.x
   ```

Issue: Ollama models fail to download

Symptoms:

```
Error: failed to pull model
```

Solutions:
1. Check Ollama service:
`bash`
```
  systemctl status ollama
```

1. Manually pull models:
   `bash`

```
   ollama pull llama3.2:latest
   ollama pull llama2:latest
   ollama pull mistral:latest
   ollama pull phi3:mini
```

2. Check disk space:
   `bash`
   `df -h`

3. Check Ollama logs:
   `bash`
   `sudo journalctl -u ollama -f`

Issue: Permission denied errors

Symptoms:

```
EACCES: permission denied, mkdir '/home/ubuntu/Sports-Bar-TV-Controller'
```

Solutions:
1. Fix directory permissions:
```bash
   sudo chown -R $USER:$USER ~/Sports-Bar-TV-Controller
   chmod -R 755 ~/Sports-Bar-TV-Controller
```

1. Run installer with correct user:
   `bash`
   `curl -sSL https://raw.githubusercontent.com/dfultonthebar/Sports-Bar-TV-Controller/main/install.sh | bash`

Issue: Database migration fails

Symptoms:

```
Error: P3009: migrate found failed migrations
```

Solutions:
1. Reset database:
```bash
   cd ~/Sports-Bar-TV-Controller
   npx prisma db push --force-reset
```

1. Rebuild knowledge base:
   `bash`
   `npm run build:kb`

Runtime Issues

Issue: Application won’t start

Symptoms:

```
Error: listen EADDRINUSE: address already in use :::3000
```

Solutions:
1. Find process using port 3000:
```bash
sudo lsof -i :3000
```

1. Kill existing process:
   ```bash
   kill -9 <PID>
   ```

2. Or change port:
   ```bash
   echo "PORT=3001" >> .env
   ```

Issue: AI features not working

Symptoms:
- AI chat returns errors
- Knowledge base queries fail
- Ollama connection errors

Solutions:
1. Check Ollama status:
```bash
systemctl status ollama
```

1. Verify models:
   ```bash
   ollama list
   ```

2. Test Ollama directly:
   ```bash
   curl http://localhost:11434/api/generate -d '{
     "model": "llama3.2:latest",
     "prompt": "Hello"
   }'
   ```

3. Rebuild knowledge base:
   ```bash
   cd ~/Sports-Bar-TV-Controller
   npm run build:kb
   ```

4. Check AI system status:
   ```bash
   curl http://localhost:3000/api/ai/status
   ```

Issue: Database errors

Symptoms:

```
PrismaClientInitializationError: Can't reach database server
```

Solutions:
1. Check database file exists:

```bash
ls -lh ~/Sports-Bar-TV-Controller/prisma/data/sports_bar.db
```

1. Verify permissions:
   ```bash
   chmod 644 ~/Sports-Bar-TV-Controller/prisma/data/sports_bar.db
   ```

2. Reset database:
   ```bash
   cd ~/Sports-Bar-TV-Controller
   npx prisma db push --force-reset
   npm run build:kb
   ```

Issue: High memory usage

Symptoms:
- System becomes slow
- Out of memory errors
- Ollama crashes

Solutions:
1. Check memory usage:

```bash
free -h
htop
```

1. Restart Ollama:
   ```bash
   sudo systemctl restart ollama
   ```

2. Limit Ollama memory:
   ```bash
   sudo systemctl edit ollama
   ```
   Add:
   ```ini
   [Service]
   Environment="OLLAMA_MAX_LOADED_MODELS=2"
   ```

3. Use smaller AI models:
   ```env
   OLLAMA_MODEL=phi3:mini
   ```

Issue: Slow AI responses

Symptoms:
- AI takes >30 seconds to respond
- Timeouts occur frequently

Solutions:
1. Use faster model:

`env`
`   OLLAMA_MODEL=phi3:mini`

1. Increase timeout:
   `env`
   `   AI_TIMEOUT=300000  # 5 minutes`

2. Check system resources:
   `bash`
   `   htop`

3. Run system benchmark:
   `bash`
   `   ./scripts/system-benchmark.sh --quick`

Network Issues

Issue: Can’t access from other devices

Symptoms:
- Works on localhost
- Doesn’t work from network

Solutions:
1. Check firewall:
`bash`
`   sudo ufw status`
`   sudo ufw allow 3000/tcp`

1. Verify application is listening on all interfaces:
   `bash`
   `   sudo netstat -tlnp | grep 3000`

2. Check HOST environment variable:
   `env`
   `   HOST=0.0.0.0`

3. Test from another device:
   `bash`
   `   curl http://[server-ip]:3000`

Issue: CORS errors in browser

Symptoms:

```
Access to fetch at 'http://...' from origin 'http://...' has been blocked by CORS
policy
```

Solutions:
1. Add origin to allowed list:

`env`
`   ALLOWED_ORIGINS=http://localhost:3000,http://192.168.1.100:3000`

1. Restart application:
   `bash`
   `   npm start`

Hardware Issues

Issue: CEC adapter not detected

Symptoms:
- CEC discovery finds no devices
- libCEC errors in logs

Solutions:
1. Check USB connection:
`bash`
`   lsusb | grep Pulse-Eight`

1. Verify libCEC installation:
   `bash`
   `   dpkg -l | grep libcec`

2. Test CEC adapter:
   `bash`
   `   cec-client -l`

3. Check permissions:
   `bash`
   `   sudo usermod -a -G dialout $USER`
   `   # Log out and back in`

Issue: Matrix switcher not responding

Symptoms:
- Can’t switch inputs
- Connection timeouts

Solutions:
1. Verify network connection:
`bash`
`   ping [matrix-ip]`

1. Test telnet connection:
   `bash`
   `   telnet [matrix-ip] [port]`

2. Check matrix configuration in Device Config

3. Review matrix logs in application

🔄 Maintenance

Regular Maintenance Tasks

Daily

- Monitor application logs
- Check system resources
- Verify AI responses

Weekly

- Review error logs
- Check disk space
- Update AI knowledge base if documentation changed

Monthly

- Update application: `./update_from_github.sh`
- Run system benchmark
- Review and rotate logs
- Check for security updates

Quarterly

- Full system backup
- Performance review
- Hardware health check
- Update documentation

Backup Procedures

Automatic Backups

The update script automatically creates backups before updates.

Backup location: `~/Sports-Bar-TV-Controller/backups/`

Manual Backup

Backup database:

```
cp ~/Sports-Bar-TV-Controller/prisma/data/sports_bar.db \
   ~/sports_bar.db.backup-$(date +%Y%m%d-%H%M%S)
```

Backup configuration:

```
tar -czf ~/sportsbar-config-$(date +%Y%m%d).tar.gz \
   ~/Sports-Bar-TV-Controller/.env \
   ~/Sports-Bar-TV-Controller/config/
```

Full backup:

```
tar -czf ~/sportsbar-full-backup-$(date +%Y%m%d).tar.gz \
   --exclude='node_modules' \
   --exclude='.next' \
   ~/Sports-Bar-TV-Controller/
```

Restore from Backup

Restore database:

```
cp ~/sports_bar.db.backup-20251007 \
   ~/Sports-Bar-TV-Controller/prisma/data/sports_bar.db
```

Restore configuration:

```
tar -xzf ~/sportsbar-config-20251007.tar.gz -C ~/
```

Full restore:

```
tar -xzf ~/sportsbar-full-backup-20251007.tar.gz -C ~/
cd ~/Sports-Bar-TV-Controller
npm install
npm run build
```

Log Management

View Logs

Application logs:

```
tail -f ~/Sports-Bar-TV-Controller/logs/app.log
```

Service logs:

```
sudo journalctl -u sportsbar-assistant -f
```

Ollama logs:

```
sudo journalctl -u ollama -f
```

Log Rotation

Configure logrotate:

```
sudo nano /etc/logrotate.d/sportsbar
```

```
/home/ubuntu/Sports-Bar-TV-Controller/logs/*.log {
    daily
    rotate 7
    compress
    delaycompress
    missingok
    notifempty
    create 0644 ubuntu ubuntu
}
```

Test logrotate:

```
sudo logrotate -f /etc/logrotate.d/sportsbar
```

Clean Old Logs

Manual cleanup:

```
cd ~/Sports-Bar-TV-Controller
./scripts/cleanup-logs.sh
```

Or use find:

```
find ~/Sports-Bar-TV-Controller/logs -name "*.log" -mtime +30 -delete
```

Update Procedures

Standard Update

```
cd ~/Sports-Bar-TV-Controller
./update_from_github.sh
```

Update with Benchmark

```
./update_from_github.sh --benchmark-quick
```

Manual Update

```
cd ~/Sports-Bar-TV-Controller
git pull origin main
npm install
npx prisma generate
npx prisma db push
npm run build
npm start
```

Performance Monitoring

System Resources

```
# CPU and memory
htop

# Disk usage
df -h
du -sh ~/Sports-Bar-TV-Controller

# Network
sudo iftop
```

Application Performance

```
# Response times
curl -w "@-" -o /dev/null -s http://localhost:3000 <<'EOF'
    time_total:  %{time_total}\n
EOF

# AI performance
curl -X POST http://localhost:3000/api/ai/chat \
  -H "Content-Type: application/json" \
  -d '{"message": "test"}' \
  -w "\nTime: %{time_total}s\n"
```

Run Benchmark

```
cd ~/Sports-Bar-TV-Controller
./scripts/system-benchmark.sh --quick
```

🔒 Security

Security Best Practices

1. **Keep system updated**
   `bash`
   `sudo apt update && sudo apt upgrade -y`

2. **Use firewall**
   `bash`
   `sudo ufw enable`
   `sudo ufw allow 22/tcp  # SSH`
   `sudo ufw allow 3000/tcp  # Application`

3. **Restrict network access**
   `bash`
   `# Allow only from local network`
   `sudo ufw allow from 192.168.1.0/24 to any port 3000`

4. **Use HTTPS in production**
   - Set up reverse proxy (Nginx)
   - Install SSL certificate (Let’s Encrypt)

5. **Secure environment variables**
   `bash`
   `chmod 600 ~/Sports-Bar-TV-Controller/.env`

6. **Regular backups**
   - Automate daily backups
   - Store backups off-site

7. **Monitor logs**
   - Review logs regularly
   - Set up alerts for errors

8. **Limit user access**
   - Use dedicated service user
   - Restrict sudo access

Securing Ollama

1. **Bind to localhost only**
   `bash`
   ```
   sudo systemctl edit ollama
   ```
   Add:
   `ini`
   ```
   [Service]
   Environment="OLLAMA_HOST=127.0.0.1:11434"
   ```

2. **Limit model access**
   `bash`
   ```
   # Remove unused models
   ollama rm model-name
   ```

3. **Monitor Ollama logs**
   `bash`
   ```
   sudo journalctl -u ollama -f
   ```

Network Security

1. **Use VPN for remote access**
   - Set up WireGuard or OpenVPN
   - Don't expose port 3000 to internet

2. **Implement rate limiting**
   - Use Nginx rate limiting
   - Protect against DDoS

3. **Enable fail2ban**
   `bash`
   ```
   sudo apt install fail2ban -y
   sudo systemctl enable fail2ban
   ```

⚡ Performance Optimization

Application Optimization

1. **Use production mode**
   `env`
   ```
   NODE_ENV=production
   ```

2. **Enable caching**
   - Configure Redis (optional)
   - Use CDN for static assets

3. **Optimize database**
   `bash`

```
   # Vacuum database
   sqlite3 ~/Sports-Bar-TV-Controller/prisma/data/sports_bar.db "VACUUM;"
```

4. **Limit AI model loading**
   bash
   ```
      sudo systemctl edit ollama
   ```
   Add:
   ini
   ```
      [Service]
      Environment="OLLAMA_MAX_LOADED_MODELS=2"
   ```

System Optimization

1. **Use SSD for database**
   - Move database to SSD
   - Update DATABASE_URL

2. **Increase swap space**
   bash
   ```
      sudo fallocate -l 4G /swapfile
      sudo chmod 600 /swapfile
      sudo mkswap /swapfile
      sudo swapon /swapfile
   ```

3. **Optimize kernel parameters**
   bash
   ```
      sudo nano /etc/sysctl.conf
   ```
   Add:
   ```
   vm.swappiness=10
      net.core.rmem_max=16777216
      net.core.wmem_max=16777216
   ```

4. **Use faster AI models**
   env
   ```
      OLLAMA_MODEL=phi3:mini
   ```

Network Optimization

1. **Use Nginx reverse proxy**
   - Enable gzip compression
   - Configure caching
   - Use HTTP/2

2. **Optimize TCP settings**
   bash
   ```
      sudo nano /etc/sysctl.conf
   ```
   Add:
   ```
   net.ipv4.tcp_fastopen=3
      net.ipv4.tcp_slow_start_after_idle=0
   ```

Monitoring Performance

1. **Run benchmarks regularly**
   `bash`
   `./scripts/system-benchmark.sh --quick`

2. **Monitor with htop**
   `bash`
   `htop`

3. **Check application metrics**
   `bash`
   `curl http://localhost:3000/api/metrics`

🗑️ Uninstall and Reinstall

Uninstalling the Application

The Sports Bar TV Controller includes a comprehensive uninstall script that safely removes the application and optionally its dependencies.

Quick Uninstall

```
# Interactive uninstall (asks for confirmation at each step)
./uninstall.sh

# Non-interactive uninstall (auto-confirms all prompts)
./uninstall.sh --yes

# One-line uninstall from GitHub
curl -sSL https://raw.githubusercontent.com/dfultonthebar/Sports-Bar-TV-Controller/
main/uninstall.sh | bash
```

Uninstall Options

Keep Dependencies (Faster Reinstall)

```
# Keep Node.js and Ollama installed
./uninstall.sh --yes --keep-nodejs --keep-ollama
```

Backup Before Uninstall

```
# Create backup of database, .env, knowledge base, and logs
./uninstall.sh --backup --yes
```

Dry Run (See What Would Be Removed)

```
# Preview what will be removed without actually removing anything
./uninstall.sh --dry-run
```

What Gets Removed

1. **Services**
   - PM2 processes
   - Systemd service
   - Ollama service (optional)

2. **Application Files**
   - Installation directory
   - Database files
   - Knowledge base
   - Logs and temporary files

3. **System Files**
   - Systemd service files
   - PM2 configuration
   - Environment files

4. **Dependencies** (optional)
   - Node.js and npm
   - Ollama and all models

Reinstalling the Application

Quick Reinstall

The fastest way to reinstall (keeps Node.js and Ollama):

```
# Local reinstall
./install.sh --reinstall --force

# One-line reinstall from GitHub
curl -sSL https://raw.githubusercontent.com/dfultonthebar/Sports-Bar-TV-Controller/
main/install.sh | bash -s -- --reinstall --force
```

Interactive Reinstall

For more control over the reinstall process:

```
# Interactive reinstall (asks for confirmation)
./install.sh --reinstall
```

Reinstall Process

1. Downloads uninstall script from GitHub
2. Runs uninstall with appropriate flags
3. Keeps Node.js and Ollama by default (faster)
4. Proceeds with normal installation
5. Rebuilds database and knowledge base
6. Restarts services

When to Reinstall

- **Corrupted Installation:** Files are damaged or missing
- **Configuration Issues:** Settings are misconfigured
- **Update Problems:** Update failed or caused issues

- **Clean Slate:** Want to start fresh with default settings
- **Testing:** Need to test installation process

Detailed Uninstall Documentation

For comprehensive uninstall documentation, including:
- All command-line options and flags
- Interactive vs non-interactive modes
- Selective uninstall procedures
- Backup and restore procedures
- Troubleshooting uninstall issues

See **UNINSTALL_GUIDE.md (./UNINSTALL_GUIDE.md)** for complete details.

📚 Additional Resources

- **README.md (./README.md)** - Quick start and overview
- **NUC_DEPLOYMENT.md (./NUC_DEPLOYMENT.md)** - Intel NUC-specific guide
- **UNINSTALL_GUIDE.md (./UNINSTALL_GUIDE.md)** - Uninstall and reinstall procedures
- **UPDATE_PROCESS.md (./UPDATE_PROCESS.md)** - Update procedures
- **BACKUP_RESTORE_GUIDE.md (./BACKUP_RESTORE_GUIDE.md)** - Backup and restore
- **ai-assistant/README.md (./ai-assistant/README.md)** - AI Code Assistant

---

🆘 Getting Help

Check Documentation

1. Read this deployment guide
2. Check troubleshooting section
3. Review installation logs

Check Logs

```
# Application logs
tail -f ~/Sports-Bar-TV-Controller/logs/app.log

# Service logs
sudo journalctl -u sportsbar-assistant -f

# Installation logs
ls -lt /tmp/sportsbar-install-*.log | head -1
```

Create an Issue

If you encounter a bug:
1. Check existing issues on GitHub
2. Create new issue with:
- Detailed description
- Steps to reproduce

- System information
- Relevant logs

✅ Deployment Checklist

Pre-Deployment

- [ ] Verify system requirements
- [ ] Check network connectivity
- [ ] Ensure sufficient disk space
- [ ] Backup existing data (if applicable)

Installation

- [ ] Run installer
- [ ] Verify Node.js installation
- [ ] Verify Ollama installation
- [ ] Check AI models downloaded
- [ ] Verify database created
- [ ] Check knowledge base built
- [ ] Test application access

Post-Installation

- [ ] Configure environment variables
- [ ] Set up systemd service (optional)
- [ ] Configure firewall
- [ ] Test from network
- [ ] Configure hardware devices
- [ ] Run system benchmark
- [ ] Set up backups
- [ ] Document configuration

Production

- [ ] Set up reverse proxy (Nginx)
- [ ] Install SSL certificate
- [ ] Configure monitoring
- [ ] Set up log rotation
- [ ] Test disaster recovery
- [ ] Document procedures
- [ ] Train users

Deployment complete! Your Sports Bar TV Controller is ready for production use. 🎉