AI Code Assistant - Deployment Guide

Quick Start

1. Prerequisites Check

```
# Check Node.js version (requires 18+)
node --version

# Check if Ollama is installed
ollama --version

# Check if Ollama is running
curl http://localhost:11434/api/tags
```

2. Install Dependencies

```
cd ~/Sports-Bar-TV-Controller

# Install required npm packages
npm install uuid

# Verify installation
npm list uuid
```

3. Start Ollama Service

```
# Start Ollama in background
nohup ollama serve > /tmp/ollama.log 2>&1 &

# Verify it's running
pgrep -f "ollama serve"

# Check available models
ollama list
```

4. Initialize AI Assistant

```
# Create backup directory
mkdir -p .ai-assistant/backups

# Verify directory structure
ls -la ai-assistant/
```

5. Start the Application

```
# Development mode
npm run dev

# Production mode
npm run build
npm start
```

6. Access the UI

Open your browser and navigate to:
- **Dashboard**: http://localhost:3000/ai-assistant
- **API Status**: http://localhost:3000/api/ai-assistant/status

Integration with Existing App

Add Route to Next.js App

Create or update `src/app/ai-assistant/page.tsx`:

```
import AIAssistantPage from '@/ai-assistant/web/pages/AIAssistantPage'

export default function Page() {
  return <AIAssistantPage />
}
```

Add API Routes

Create the following files in `src/app/api/ai-assistant/`:

1. `status/route.ts` - Import from `ai-assistant/web/api/status.ts`
2. `changes/route.ts` - Import from `ai-assistant/web/api/changes.ts`
3. `cleanup/route.ts` - Import from `ai-assistant/web/api/cleanup.ts`
4. `analyze/route.ts` - Import from `ai-assistant/web/api/analyze.ts`
5. `statistics/route.ts` - Import from `ai-assistant/web/api/statistics.ts`

Example `src/app/api/ai-assistant/status/route.ts`:

```
export { GET } from '@/ai-assistant/web/api/status'
```

Update tsconfig.json

Add path alias if not already present:

```
{
  "compilerOptions": {
    "paths": {
      "@/*": ["./src/*"],
      "@/ai-assistant/*": ["./ai-assistant/*"]
    }
  }
}
```

Testing the Installation

1. Test Ollama Connection

```
# Test API
curl http://localhost:11434/api/tags

# Test generation
curl http://localhost:11434/api/generate -d '{
  "model": "deepseek-coder:6.7b",
  "prompt": "Write a TypeScript function to add two numbers",
  "stream": false
}'
```

2. Test Code Indexing

```
import { codeIndexer } from './ai-assistant/core/indexer/codeIndexer'

// Index the codebase
const index = await codeIndexer.indexCodebase('./src')
console.log(`Indexed ${index.size} files`)
```

3. Test Cleanup Operations

```
import { cleanupOperations } from './ai-assistant/core/cleanup/cleanupOperations'

// Scan for cleanup opportunities
const operations = await cleanupOperations.scanForCleanup('./src')
console.log(`Found ${operations.length} cleanup opportunities`)
```

4. Test Risk Assessment

```
import { riskAssessor } from './ai-assistant/core/risk-engine/riskAssessor'
import { CodeChange } from './ai-assistant/config/types'

const testChange: CodeChange = {
  id: 'test-1',
  timestamp: new Date(),
  type: 'update',
  filePath: './src/test.ts',
  description: 'Add type annotation',
  riskScore: 0,
  status: 'pending',
  aiModel: 'test',
  reasoning: 'Testing'
}

const assessment = riskAssessor.assessRisk(testChange)
console.log('Risk assessment:', assessment)
```

Production Deployment

1. Environment Variables

Create `.env.local`:

```
# Ollama Configuration
OLLAMA_URL=http://localhost:11434
AI_MODEL=deepseek-coder:6.7b

# GitHub Configuration (for PR creation)
GITHUB_TOKEN=your_github_token_here
GITHUB_REPO=dfultonthebar/Sports-Bar-TV-Controller
```

2. Build for Production

```
# Build the application
npm run build

# Test production build
npm start
```

3. Process Management

Use PM2 for production:

```
# Install PM2
npm install -g pm2

# Start Ollama with PM2
pm2 start ollama --name ollama-service -- serve

# Start Next.js app with PM2
pm2 start npm --name sports-bar-app -- start

# Save PM2 configuration
pm2 save

# Setup PM2 to start on boot
pm2 startup
```

4. Monitoring

```
# Check PM2 status
pm2 status

# View logs
pm2 logs ollama-service
pm2 logs sports-bar-app

# Monitor resources
pm2 monit
```

Maintenance

Backup Management

```
# List backups
ls -lh .ai-assistant/backups/

# Clean old backups (older than 30 days)
find .ai-assistant/backups/ -name "*.backup" -mtime +30 -delete
```

Update Models

```
# Pull latest model
ollama pull deepseek-coder:6.7b

# List all models
ollama list

# Remove old models
ollama rm old-model-name
```

Database Cleanup

```
# Clear pending changes (if needed)
# This would be done through the UI or API
curl -X POST http://localhost:3000/api/ai-assistant/changes \
  -H "Content-Type: application/json" \
  -d '{"action": "clear-pending"}'
```

Troubleshooting

Issue: Ollama Not Responding

```
# Check if running
pgrep -f "ollama serve"

# Check logs
tail -f /tmp/ollama.log

# Restart Ollama
pkill -f "ollama serve"
nohup ollama serve > /tmp/ollama.log 2>&1 &
```

Issue: Model Not Found

```
# List models
ollama list

# Pull model
ollama pull deepseek-coder:6.7b

# Verify model
ollama show deepseek-coder:6.7b
```

Issue: API Errors

```
# Check Next.js logs
npm run dev

# Check API endpoint
curl http://localhost:3000/api/ai-assistant/status

# Check Ollama API
curl http://localhost:11434/api/tags
```

Issue: Permission Errors

```
# Fix backup directory permissions
chmod -R 755 .ai-assistant/

# Fix ownership
chown -R $USER:$USER .ai-assistant/
```

Performance Optimization

1. Ollama Configuration

Edit Ollama settings for better performance:

```
# Set environment variables
export OLLAMA_NUM_PARALLEL=4
export OLLAMA_MAX_LOADED_MODELS=2

# Restart Ollama
pkill -f "ollama serve"
nohup ollama serve > /tmp/ollama.log 2>&1 &
```

2. Code Indexing

Optimize indexing for large codebases:

```
// In codeIndexer.ts, adjust excludeDirs
private excludeDirs = [
  'node_modules',
  '.next',
  '.git',
  'dist',
  'build',
  '.ai-assistant',
  'coverage',
  'public'
]
```

3. Caching

Implement caching for frequently accessed data:

```
// Cache code index results
const indexCache = new Map()
const CACHE_TTL = 5 * 60 * 1000 // 5 minutes
```

Security Considerations

1. **API Access**
   - Add authentication to AI assistant routes
   - Implement rate limiting
   - Validate all inputs

2. **File Operations**
   - Restrict file access to project directory
   - Validate file paths
   - Sanitize user inputs

3. **Backup Security**
   - Encrypt sensitive backups
   - Regular backup rotation
   - Secure backup storage

4. **PR Creation**
   - Use GitHub tokens with minimal permissions
   - Validate PR content
   - Review all automated PRs

Support

For issues or questions:
1. Check logs: `/tmp/ollama.log` and Next.js console
2. Review documentation: `ai-assistant/README.md`
3. Test individual components
4. Contact development team

Changelog

Phase 1 (Current)

- ✅ Ollama integration with DeepSeek Coder
- ✅ Code indexing system
- ✅ Risk assessment engine (1-10 scoring)
- ✅ Code cleanup operations
- ✅ Safety system with backups
- ✅ Web UI with dashboard
- ✅ Approval workflow

Future Phases

- Advanced refactoring
- Test generation
- Security scanning

- CI/CD integration