🚀 Quick Start Guide - AI Code Assistant

✅ What's Been Completed

Part 1: TypeScript Error Fixed ✅

- Fixed type inference issue in `downloadManual.ts`
- PR #89 created and ready for review
- Build error resolved

Part 2: AI Code Assistant Built ✅

- Complete Phase 1 implementation
- PR #90 created with 25 new files
- All features functional and tested

🎯 Immediate Next Steps

1. Review and Merge PRs

PR #89 - TypeScript Fix (Low Risk)

```
# Review at: https://github.com/dfultonthebar/Sports-Bar-TV-Controller/pull/89
# This is a simple one-line fix - safe to merge
```

PR #90 - AI Assistant (New Feature)

```
# Review at: https://github.com/dfultonthebar/Sports-Bar-TV-Controller/pull/90
# This adds new functionality without modifying existing code
```

2. Deploy the AI Assistant

Step 1: Verify Ollama is Running

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

# If not running, start it
nohup ollama serve > /tmp/ollama.log 2>&1 &

# Verify the model is available
ollama list
# Should show: deepseek-coder:6.7b
```

Step 2: Pull Latest Code (After Merging PRs)

```
cd ~/Sports-Bar-TV-Controller
git checkout main
git pull origin main
```

Step 3: Install Dependencies

```
npm install
# uuid package should already be installed
```

Step 4: Start the Application

```
# Development mode
npm run dev

# Or production mode
npm run build
npm start
```

Step 5: Access the AI Assistant

```
# Open in browser:
http://localhost:3000/ai-assistant
```

🧪 Test the System

Quick Test

```
# Run the test script
npx ts-node ai-assistant/test-system.ts
```

Expected Output

```
🚀 Testing AI Code Assistant System

1️⃣ Testing Ollama Connection...
✅ Ollama is available
   Available models: deepseek-coder:6.7b

2️⃣ Testing Code Indexing...
✅ Indexed 150 files

3️⃣ Testing Risk Assessment...
✅ Risk assessment completed
   Score: 10/10
   Category: safe
   Recommendation: auto-apply

4️⃣ Testing Safety System...
✅ Safety system initialized

5️⃣ Testing Change Manager...
✅ Change manager initialized

6️⃣ Testing Cleanup Operations...
✅ Found 5 cleanup opportunities

7️⃣ Testing AI Code Generation...
✅ AI code generation successful

✨ System test complete!
```

📖 Using the AI Assistant

Via Web UI

1. **Dashboard** - View statistics and system status
   - Navigate to: http://localhost:3000/ai-assistant
   - See pending changes, applied changes, and system health

2. **Pending Changes** - Review and approve changes
   - Click “Pending Changes” tab
   - Review each change with diff
   - Approve or reject

3. **History** - View all past changes
   - Click “History” tab
   - See complete audit trail
   - Access PR links

Via Code

Example 1: Scan for Cleanup

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

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

Example 2: Analyze Code

```
import { ollamaService } from './ai-assistant/services/ollamaService'

const analysis = await ollamaService.analyzeCode(code, filePath)
console.log(analysis)
```

Example 3: Propose Change

```
import { changeManager } from './ai-assistant/services/changeManager'

await changeManager.initialize()

const { change, assessment } = await changeManager.proposeChange(
  './src/file.ts',
  'update',
  'Fix type annotation',
  newContent,
  'deepseek-coder',
  'Adding explicit type'
)

console.log(`Risk Score: ${assessment.score}/10`)
```

🎓 Understanding Risk Scores

Score 10 - Safe (Auto-apply)

- Lint fixes
- Remove unused imports
- Add comments
- Type annotations

Action: Changes are applied automatically

Score 7-9 - Medium (Create PR)

- Code updates
- Small refactoring
- API changes
- New files

Action: PR is created for review

Score 1-6 - High (Require Approval)

- Config changes
- Database changes
- Auth code
- File deletions
- Large refactoring

Action: Manual approval required

🛡️ Safety Features

1. Automatic Backups

Every change creates a backup in `.ai-assistant/backups/`

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

# Restore from backup (via UI or code)
```

2. Git Integration

Changes create feature branches automatically

```
# Example branch name
ai-assistant/update-1728234567
```

3. PR Creation

Medium-risk changes create PRs with full context

4. Rollback

One-click rollback from the UI or code

```
await changeManager.rollbackChange(changeId)
```

📊 Monitoring

Check System Status

```
# Via API
curl http://localhost:3000/api/ai-assistant/status

# Via UI
# Navigate to dashboard
```

View Statistics

```
# Via API
curl http://localhost:3000/api/ai-assistant/statistics

# Via UI
# Dashboard shows real-time stats
```

Check Logs

```
# Ollama logs
tail -f /tmp/ollama.log

# Next.js logs
# Visible in terminal where npm run dev is running
```

🔧 Configuration

Adjust Risk Thresholds

Edit `ai-assistant/config/config.ts`:

```
export const AI_ASSISTANT_CONFIG = {
  riskThresholds: {
    safe: 10,      // Change to adjust auto-apply threshold
    medium: 7,     // Change to adjust PR creation threshold
    high: 1        // Change to adjust approval requirement
  },

  autoApplyThreshold: 10,  // Only auto-apply score 10
  enableAutoBackup: true,  // Always create backups
  enablePRCreation: true   // Create PRs for medium risk
}
```

Change AI Model

```
# Pull a different model
ollama pull codellama:7b

# Update config
# Edit ai-assistant/config/config.ts
model: 'codellama:7b'
```

🚨 Troubleshooting

Issue: Ollama Not Responding

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

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

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

Issue: Model Not Found

```
# List models
ollama list

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

Issue: UI Not Loading

```
# Check if Next.js is running
# Should see output in terminal

# Restart
npm run dev

# Check port
lsof -i :3000
```

Issue: Permission Errors

```
# Fix backup directory
chmod -R 755 .ai-assistant/
chown -R $USER:$USER .ai-assistant/
```

📚 Documentation

Main Docs

- **README.md**: Complete usage guide
- **DEPLOYMENT.md**: Detailed deployment instructions
- **EXAMPLES.md**: 10 usage examples
- **AI_ASSISTANT_SUMMARY.md**: This summary

Quick Links

- PR #89: https://github.com/dfultonthebar/Sports-Bar-TV-Controller/pull/89
- PR #90: https://github.com/dfultonthebar/Sports-Bar-TV-Controller/pull/90

- Ollama Docs: https://ollama.ai/docs
- DeepSeek Coder: https://ollama.ai/library/deepseek-coder

✅ Checklist

Before Using

- [ ] Ollama is running
- [ ] DeepSeek Coder model is pulled
- [ ] Dependencies installed (npm install)
- [ ] Application is running (npm run dev)
- [ ] Can access UI at localhost:3000/ai-assistant

First Use

- [ ] Run test script (npx ts-node ai-assistant/test-system.ts)
- [ ] Check dashboard loads
- [ ] Verify Ollama status shows “Online”
- [ ] Test a simple cleanup operation
- [ ] Review a pending change

Regular Use

- [ ] Monitor pending changes
- [ ] Review and approve/reject changes
- [ ] Check change history
- [ ] Clean old backups periodically
- [ ] Monitor system resources

🎉 Success!

You now have a fully functional AI Code Assistant that can:
- ✅ Analyze your codebase
- ✅ Suggest improvements
- ✅ Clean up code automatically
- ✅ Assess risk of changes
- ✅ Create PRs for review
- ✅ Maintain safety with backups
- ✅ Track all changes

Next Steps

1. Merge the PRs
2. Test the system
3. Start using it for code improvements
4. Provide feedback for Phase 2

💬 Need Help?

1. Check the documentation in `ai-assistant/` directory
2. Review the examples in `EXAMPLES.md`
3. Run the test script to diagnose issues
4. Check logs for error messages

Status: ✅ Ready to Deploy

Date: October 6, 2025

Version: 1.0.0 (Phase 1)