# Channel Preset Quick Access Implementation

## Overview

This document describes the implementation of channel preset quick access buttons on the bartender remote interface.

## Features Implemented

### 1. Backend API Enhancements

#### New API Endpoints

**GET /api/channel-presets/by-device**
- Fetches presets filtered by device type (cable or directv)
- Returns presets ordered by usage count (AI-sorted) or alphabetically
- Query parameters:
- `deviceType`: 'cable' | 'directv' (required)
- Response includes:
- List of presets
- Ordering method (usage-based or alphabetical)
- Usage data availability flag

**POST /api/channel-presets/update-usage**
- Updates usage tracking when a preset is clicked
- Increments `usageCount` field
- Updates `lastUsed` timestamp
- Request body:
- `presetId`: string (required)

**GET /api/cron/init**
- Initializes all cron jobs (called on app startup)
- Returns list of active cron jobs and their schedules

#### Services Created

**presetReorderService.ts** ( `src/services/presetReorderService.ts` )
- `reorderAllPresets()`: Reorders all presets based on usage count
- `getUsageStatistics()`: Returns comprehensive usage statistics
- `checkReorderingNeeded()`: Checks if reordering is needed for a device type

**presetCronService.ts** ( `src/services/presetCronService.ts` )
- `initializePresetCronJob()`: Sets up monthly cron job
- `stopPresetCronJob()`: Stops the cron job
- `triggerManualReorder()`: Manually triggers reordering for testing
- Schedule: Runs at 3:00 AM on the 1st of each month
- Timezone: America/New_York (configurable)

## 2. Frontend UI Implementation

### New Component: ChannelPresetGrid

**Location**: `src/components/ChannelPresetGrid.tsx`

**Features**:
- Displays preset buttons in a responsive grid (2-3 columns)
- Shows top 6 presets by default, expandable to show all
- Each button displays:
- Channel name
- Channel number
- Usage count badge (if > 0)
- Visual indicators:
- AI sorting badge when usage data exists
- Loading state
- Empty state with helpful message
- Touch-friendly design for bartenders
- Automatic usage tracking on click

**Props**:
- `deviceType`: 'cable' | 'directv'
- `onPresetClick`: Callback function when preset is clicked
- `maxVisible`: Number of presets to show initially (default: 6)

### Modified Component: BartenderRemoteControl

**Location**: `src/components/BartenderRemoteControl.tsx`

**Changes**:
1. Added import for `ChannelPresetGrid` component
2. Added `handlePresetClick()` function:
- Validates input selection
- Sends channel change command
- Updates usage tracking
- Logs operation
3. Added `sendDirecTVChannelChange()` helper function
4. Added `getSelectedInputDeviceType()` function:
- Determines if selected input is cable or directv
- Returns null for other input types
5. Integrated `ChannelPresetGrid` component:
- Appears below channel controls
- Only shown when Cable Box or DirecTV input is selected
- Automatically detects device type

## 3. Cron Job Initialization

**Auto-initialization**: `src/lib/initCronJobs.ts`
- Automatically initializes cron jobs when the server starts
- Prevents duplicate initialization
- Server-side only (checks for `window` object)

**Manual initialization**: Available via API endpoint
- GET `/api/cron/init` - Initialize cron jobs manually
- Useful for testing or restarting jobs

### 4. AI Re-ordering Logic

**Initial Ordering**: Alphabetical by channel name
- When presets are first created, they have `order = 0` and `usageCount = 0`
- API returns them sorted alphabetically by name

**AI Re-ordering**: Monthly automatic re-ordering
- Cron job runs at 3:00 AM on the 1st of each month
- Reorders presets by `usageCount` (descending)
- Updates the `order` field in database
- Separate ordering for cable and directv presets
- Ties broken by alphabetical name

**Usage Tracking**:
- Every preset click increments `usageCount`
- Updates `lastUsed` timestamp
- Happens in background (non-blocking)
- Failures don't affect user experience

## Database Schema

The `ChannelPreset` table already includes all necessary fields:

```
model ChannelPreset {
  id            String    @id @default(cuid())
  name          String    // User-friendly name
  channelNumber String    // Channel number to tune to
  deviceType    String    // "cable" or "directv"
  order         Int       @default(0)  // Display order
  isActive      Boolean   @default(true)
  usageCount    Int       @default(0)  // Usage tracking
  lastUsed      DateTime? // Last used timestamp
  createdAt     DateTime  @default(now())
  updatedAt     DateTime  @updatedAt

  @@index([deviceType, order])
  @@index([isActive])
  @@index([usageCount])
}
```

## Integration Flow

### User Flow:

1. Bartender selects a TV input (Cable Box or DirecTV)
2. Preset grid appears below channel controls
3. Grid shows top 6 presets (or all if expanded)
4. Bartender clicks a preset button
5. System:
   - Sends channel change command to device
   - Updates usage tracking in background
   - Shows visual feedback
   - Logs operation

### AI Re-ordering Flow:

1. Cron job triggers monthly (1st of month, 3:00 AM)
2. Service fetches all active presets by device type
3. Sorts by usage count (descending), then name (ascending)
4. Updates `order` field for each preset
5. Next time presets are fetched, they appear in new order

## Testing

### Manual Testing:

1. **Test Preset Display**:
   - Select a Cable Box or DirecTV input
   - Verify preset grid appears
   - Verify presets are displayed correctly

2. **Test Preset Click**:
   - Click a preset button
   - Verify channel changes on device
   - Verify usage count increments (check database or statistics API)

3. **Test Usage Tracking**:
   - Click multiple presets
   - Call GET `/api/channel-presets/statistics`
   - Verify usage counts are tracked

4. **Test Manual Reordering**:
   - Call POST `/api/channel-presets/reorder`
   - Verify presets are reordered by usage
   - Refresh preset grid and verify new order

5. **Test Cron Job**:
   - Call GET `/api/cron/init` to verify initialization
   - Check server logs for cron job messages

### API Testing:

```
# Get presets for cable devices
curl http://localhost:3000/api/channel-presets/by-device?deviceType=cable

# Get presets for directv devices
curl http://localhost:3000/api/channel-presets/by-device?deviceType=directv

# Update usage for a preset
curl -X POST http://localhost:3000/api/channel-presets/update-usage \
  -H "Content-Type: application/json" \
  -d '{"presetId": "preset_id_here"}'

# Get usage statistics
curl http://localhost:3000/api/channel-presets/statistics

# Manually trigger reordering
curl -X POST http://localhost:3000/api/channel-presets/reorder

# Initialize cron jobs
curl http://localhost:3000/api/cron/init
```

## Configuration

### Cron Schedule

To change the cron schedule, edit `src/services/presetCronService.ts`:

```
// Current: 3:00 AM on 1st of each month
cronJob = cron.schedule('0 3 1 * *', async () => {
  // ...
})

// Examples:
// Daily at 3 AM: '0 3 * * *'
// Weekly on Monday at 3 AM: '0 3 * * 1'
// Every 6 hours: '0 */6 * * *'
```

### Timezone

To change the timezone, edit `src/services/presetCronService.ts`:

```
{
  scheduled: true,
  timezone: 'America/New_York' // Change to your timezone
}
```

### Max Visible Presets

To change the default number of visible presets, edit `src/components/BartenderRemoteControl.tsx`:

```
<ChannelPresetGrid
  deviceType={getSelectedInputDeviceType()!}
  onPresetClick={handlePresetClick}
  maxVisible={6} // Change this number
/>
```

# Dependencies

All required dependencies are already installed:
- `node-cron`: ^4.2.1 (for cron jobs)
- `@types/node-cron`: ^3.0.11 (TypeScript types)

# Files Created/Modified

## New Files:

- `src/services/presetReorderService.ts` - Preset reordering logic
- `src/services/presetCronService.ts` - Cron job management
- `src/app/api/channel-presets/update-usage/route.ts` - Usage tracking API
- `src/app/api/channel-presets/by-device/route.ts` - Device-filtered preset API
- `src/app/api/cron/init/route.ts` - Cron initialization API
- `src/components/ChannelPresetGrid.tsx` - Preset grid component
- `src/lib/initCronJobs.ts` - Auto-initialization utility
- `CHANNEL_PRESET_QUICK_ACCESS.md` - This documentation

## Modified Files:

- `src/components/BartenderRemoteControl.tsx` - Added preset grid integration

# Future Enhancements

Potential improvements for future iterations:
1. **Real-time Adaptive Reordering**: Reorder presets in real-time based on usage patterns
2. **Time-based Presets**: Show different presets based on time of day or day of week
3. **Favorite Presets**: Allow bartenders to mark presets as favorites
4. **Preset Categories**: Group presets by sport or channel type
5. **Multi-device Presets**: Support presets that work across multiple device types
6. **Preset Analytics Dashboard**: Visualize usage patterns and trends
7. **Voice Control**: Add voice commands for preset selection
8. **Custom Preset Icons**: Allow custom icons for each preset

# Troubleshooting

## Presets Not Appearing

- Verify presets exist in database for the device type
- Check that presets have `isActive = true`
- Verify device type detection is working correctly

## Usage Tracking Not Working

- Check browser console for API errors
- Verify `/api/channel-presets/update-usage` endpoint is accessible
- Check database permissions

## Cron Job Not Running

- Check server logs for initialization messages
- Verify cron syntax is correct

- Ensure server timezone is configured correctly
- Test manual reordering via API

### Channel Changes Not Working

- Verify device is configured correctly
- Check device IP address and port
- Test channel changes using regular number pad
- Check device logs for errors

## Support

For issues or questions:
1. Check server logs for error messages
2. Test API endpoints directly using curl
3. Verify database schema matches expected structure
4. Check that all dependencies are installed
5. Review this documentation for configuration options