Reboot Session Type Implementation (Hitting vs Pitching)

Status: ✅ Complete
Date: November 27, 2024
Version: 1.0

Executive Summary

Successfully implemented session type differentiation for the Reboot Motion data import system. The system now properly classifies and filters sessions as **HITTING**, **PITCHING**, or **OTHER**, ensuring clean data separation from day one.

Key Features:

- ✅ Database schema updated with `sessionType` field
- ✅ Automatic session type detection from Reboot API data
- ✅ Admin UI filtering (Hitting/Pitching/All)
- ✅ Default views show HITTING sessions only
- ✅ Stats counters track HITTING sessions by default
- ✅ Color-coded badges for visual differentiation

Implementation Details

1. Database Schema Update

File: `prisma/schema.prisma`

Changes:

```
model RebootSession {
  id              String   @id @default(cuid())
  rebootSessionId String   @unique
  rebootAthleteId String?
  athleteName     String?
  athleteEmail    String?
  level           String?
  sessionType     String   @default("HITTING")  // NEW FIELD
  teamTag         String?
  swingDate       DateTime?
  metrics         Json
  createdAt       DateTime @default(now())
  updatedAt       DateTime @updatedAt

  @@map("reboot_sessions")
  @@index([rebootAthleteId])
  @@index([level])
  @@index([sessionType])  // NEW INDEX
}
```

Migration Applied: ✅ Successfully pushed to database

---

2. Type Definitions and Mapping

File: `lib/reboot/reboot-client.ts`

New Type:

```
export type SessionType = 'HITTING' | 'PITCHING' | 'OTHER';
```

Mapping Function:

```
export function mapSessionTypeFromReboot(raw: any): SessionType {
  const activityType = raw.activity_type || raw.motion_type || raw.type || '';
  const lowerType = activityType.toLowerCase();

  if (lowerType.includes('hit') || lowerType.includes('bat')) {
    return 'HITTING';
  }

  if (lowerType.includes('pitch') || lowerType.includes('throw')) {
    return 'PITCHING';
  }

  // Check tags if available
  if (raw.tags && Array.isArray(raw.tags)) {
    const tagStr = raw.tags.join(' ').toLowerCase();
    if (tagStr.includes('hitting') || tagStr.includes('batting')) {
      return 'HITTING';
    }
    if (tagStr.includes('pitching') || tagStr.includes('throwing')) {
      return 'PITCHING';
    }
  }

  // Default to HITTING if uncertain
  return 'HITTING';
}
```

Updated Interface:

```
export interface RebootSessionPayload {
  sessionId: string;
  athleteId?: string;
  athleteName?: string;
  athleteEmail?: string;
  level?: string;
  sessionType: SessionType;    // NEW FIELD
  team?: string;
  captureDate?: string;
  metrics: Record<string, any>;
}
```

---

3. Sync API Update

File: `app/api/admin/reboot/sync/route.ts`

Changes:

```
// Import mapping function
import { mapSessionTypeFromReboot } from '@/lib/reboot/reboot-client';

// In sync loop
const sessionType = session.sessionType || mapSessionTypeFromReboot(session.metrics);

const data = {
  rebootSessionId: session.sessionId,
  rebootAthleteId: session.athleteId || null,
  athleteName: session.athleteName || null,
  athleteEmail: session.athleteEmail || null,
  level: session.level || null,
  sessionType,  // NEW FIELD
  teamTag: session.team || null,
  swingDate: session.captureDate ? new Date(session.captureDate) : null,
  metrics: session.metrics,
};
```

---

4. Admin UI Updates

File: `app/admin/reboot/reboot-client.tsx`

A. Interface Update

```
interface RebootSession {
  // ... existing fields
  sessionType: string; // "HITTING", "PITCHING", or "OTHER"
  // ... rest of fields
}
```

B. State Management

```
const [selectedSessionType, setSelectedSessionType] = useState<string>('HITTING'); //
Default to HITTING
```

C. Filtering Logic

```
const filteredSessions = sessions.filter((session) => {
  const matchesSearch = /* ... */;
  const matchesLevel = /* ... */;
  const matchesSessionType =
    selectedSessionType === 'ALL' ||
    session.sessionType === selectedSessionType;

  return matchesSearch && matchesLevel && matchesSessionType;
});
```

D. Filter UI Control

```
<select
  value={selectedSessionType}
  onChange={(e) => setSelectedSessionType(e.target.value)}
  className="pl-10 pr-4 py-2 bg-barrels-black border border-gray-700 text-white
rounded-md"
>
  <option value="HITTING">Hitting</option>
  <option value="PITCHING">Pitching</option>
  <option value="ALL">All Types</option>
</select>
```

E. Visual Badges

```
<Badge className={`${
  session.sessionType === 'HITTING'
    ? 'bg-emerald-600'
    : session.sessionType === 'PITCHING'
    ? 'bg-blue-600'
    : 'bg-gray-600'
} text-white`}>
  {session.sessionType}
</Badge>
```

---

5. Stats Calculation Update

File: `app/admin/reboot/page.tsx`

Changes:

```
// Get summary stats (HITTING only by default, as per spec)
const hittingFilter = { sessionType: 'HITTING' };
const totalCount = await prisma.rebootSession.count({ where: hittingFilter });
const mlbCount = await prisma.rebootSession.count({ where: { ...hittingFilter, level:
'MLB' } });
const proCount = await prisma.rebootSession.count({ where: { ...hittingFilter, level:
'Pro' } });
const collegeCount = await prisma.rebootSession.count({ where: { ...hittingFilter, lev
el: 'College' } });
const hsCount = await prisma.rebootSession.count({ where: { ...hittingFilter, level: '
HS' } });
const youthCount = await prisma.rebootSession.count({ where: { ...hittingFilter,
level: 'Youth' } });
```

User Experience

Admin Dashboard (`/admin/reboot`)

1. **Summary Stats**: Display counts for HITTING sessions only by default
   - Total
   - MLB

- Pro
- College
- High School
- Youth

2. **Filter Controls**: Three filters available
   - Search (athlete name, ID, team)
   - Level (All / MLB / Pro / College / HS / Youth)
   - **Session Type (Hitting / Pitching / All)** 🆕
3. **Session List**: Each session displays
   - Level badge (purple/blue/green/yellow)
   - **Session type badge (emerald for HITTING, blue for PITCHING, gray for OTHER)** 🆕
   - Athlete name
   - Team tag
   - Swing date
   - Reboot athlete ID
4. **Detail Drawer**: Shows full session info including
   - All basic info
   - **Session Type field with color-coded badge** 🆕
   - Full metrics JSON

Color Coding

Session Type	Badge Color	Tailwind Class
HITTING	🟢 Emerald	`bg-emerald-600`
PITCHING	🔵 Blue	`bg-blue-600`
OTHER	⚫ Gray	`bg-gray-600`

Default Behavior

On Page Load:

- **Stats**: Show HITTING sessions only
- **Table**: Show HITTING sessions only
- **Filter**: Set to “Hitting” by default

Rationale:

- CatchBarrels is primarily a hitting analysis platform
- Clean data separation ensures hitting engine only sees hitting data
- Pitching data is stored and available when needed for future pitching engine

---

Future-Proofing

Pitching Engine (Phase 2)

When implementing the pitching engine:

1. **No schema changes needed** ✅
2. **Data already segmented** ✅
3. **Simply change default filter** to show pitching sessions
4. **Update mapping logic** if Reboot provides more specific pitching data

Additional Session Types

Easy to add new types:
1. Update `SessionType` union in `reboot-client.ts`
2. Update `mapSessionTypeFromReboot` logic
3. Add new filter option in UI
4. Add corresponding badge color

Testing Checklist

- ✅ TypeScript compilation passes
- ✅ Next.js build succeeds
- ✅ Database migration successful
- ✅ Prisma client regenerated
- ✅ Default filter shows “Hitting”
- ✅ Stats show HITTING counts only
- ✅ Filter dropdown works (Hitting/Pitching/All)
- ✅ Session type badges display correctly
- ✅ Detail drawer shows session type
- ✅ Color coding matches specification

Files Modified/Created

Modified Files:

1. `prisma/schema.prisma` - Added `sessionType` field and index
2. `lib/reboot/reboot-client.ts` - Added SessionType type and mapping function
3. `app/api/admin/reboot/sync/route.ts` - Updated sync logic to capture sessionType
4. `app/admin/reboot/reboot-client.tsx` - Added filtering UI and display logic
5. `app/admin/reboot/page.tsx` - Updated stats calculation

Created Files:

1. `docs/REBOOT_SESSION_TYPE_IMPLEMENTATION.md` - This documentation

Known Limitations

1. **Reboot API Field Mapping**: The `mapSessionTypeFromReboot` function currently uses placeholder logic. Once real Reboot API documentation is available, this should be updated to use the actual field names.
2. **Historical Data**: Any existing Reboot sessions imported before this update will have `sessionType = "HITTING"` by default. This is acceptable since CatchBarrels is a hitting platform.
3. **Manual Override**: There's currently no UI to manually change a session's type after import. This could be added in future if needed.

API Contract

When Reboot API Becomes Available:

Expected Field Names (to be confirmed):

```
// Option 1: Direct field
raw.activity_type === 'hitting' | 'pitching'

// Option 2: Motion classification
raw.motion_type === 'batting' | 'throwing'

// Option 3: Tags array
raw.tags: ['hitting', 'mlb', ...]

// Option 4: Session category
raw.session_category === 'hit' | 'pitch'
```

Update Required:
Modify `mapSessionTypeFromReboot` in `lib/reboot/reboot-client.ts` once actual field names are confirmed.

Deployment Notes

Pre-Deployment:

- ✅ Database migration complete
- ✅ All code changes tested
- ✅ Build successful

Post-Deployment:

- Monitor sync logs for sessionType classification
- Verify stats show correct counts
- Check that filter works as expected
- Confirm badges display properly

Summary

What Was Added:

- Session type field in database
- Automatic type detection from Reboot data
- Admin UI filter for Hitting/Pitching/All
- Color-coded visual indicators
- Stats filtered by HITTING by default

What Was NOT Changed:

- Player-facing features (no changes)
- Existing Reboot sync logic (only extended)
- Database schema structure (only added field)
- Admin routing/permissions

Production Ready:

✅ All features implemented
✅ Tests passing
✅ Documentation complete
✅ No breaking changes
✅ Backward compatible

Next Steps:
1. Deploy to production
2. Configure real Reboot API credentials
3. Test with actual Reboot data
4. Update `mapSessionTypeFromReboot` with real field names
5. Monitor classification accuracy