This technical implementation plan provides the necessary details for an engineer to integrate Whoop API data into the EchoVault ecosystem.

Technical Specification: Whoop Cloud-to-Cloud Integration

1. Project Overview

Integrate the Whoop V2 API to provide health data synchronization for web and PWA users, bypassing native-only limitations of HealthKit and Google Fit. This integration uses a cloud-to-cloud OAuth 2.0 flow managed by the `relay-server`.

---

2. Backend Requirements (`relay-server`)

A. Configuration Updates

Modify relay-server/src/config/index.ts to include Whoop environment variables:

```TypeScript
export const config = {
  // ... existing config
  whoopClientId: process.env.WHOOP_CLIENT_ID || '',
  whoopClientSecret: process.env.WHOOP_CLIENT_SECRET || '',
  whoopRedirectUri: process.env.WHOOP_REDIRECT_URI ||
'https://your-app.com/auth/whoop/callback',
} as const;
```

B. Authentication & Token Management

1. **OAuth Flow**: Implement endpoints to handle the authorization code grant.
2. **Storage**: Securely store `access_token` and `refresh_token` in Firestore under the user's profile.
3. **Automatic Refresh**: Implement logic within `relay/sessionManager.ts` to check token expiration and use the refresh token before making data requests.

C. Proxy Endpoints

Create a new controller to fetch and normalize Whoop data:

- `GET /health/whoop/recovery`: Fetches HRV and recovery scores.
- `GET /health/whoop/sleep`: Fetches sleep stages and duration.

---

3. Frontend Service Implementation

A. Create src/services/health/whoop.js

This service acts as the client-side interface for Whoop data.

```JavaScript
```

```javascript
import { cacheHealthData } from './platformHealth';

/**
 * Fetch Whoop summary via Relay Server
 */
export const getWhoopSummary = async (date = new Date()) => {
  try {
    const response = await
fetch(`${RELAY_URL}/health/whoop/summary?date=${date.toISOString()}`);
    const data = await response.json();

    const summary = {
      available: true,
      source: 'whoop',
      date: date.toISOString().split('T')[0],
      sleep: {
        totalHours: data.sleep.duration / 3600000, // Convert ms to hours
        quality: data.sleep.score >= 80 ? 'good' : 'fair'
      },
      hrv: {
        average: data.recovery.hrv,
        stressIndicator: data.recovery.score < 33 ? 'high' : 'low'
      },
      heartRate: { resting: data.recovery.rhr },
      queriedAt: new Date().toISOString()
    };

    await cacheHealthData(summary); // Cache for offline web access
    return summary;
  } catch (error) {
    return { available: false, source: 'whoop', error: error.message };
  }
};
```

B. Update src/services/health/healthDataService.js

Integrate Whoop into the unified data fetcher:

JavaScript

```javascript
export const getHealthSummary = async (date = new Date()) => {
  const strategy = await getHealthDataStrategy();

  // Prioritize Whoop if the user has linked their account
  if (userHasLinkedWhoop()) {
    return await getWhoopSummary(date);
  }

  switch (strategy.strategy) {
    case 'healthkit': return await getHealthKitSummary(date);
    case 'googlefit': return await getGoogleFitSummary(date);
    // ...
  }
};
```

---

4. Data Mapping & Analysis Integration

Ensure Whoop data maps correctly to existing analysis engines:

- **Mood Correlation**: Whoop's `hrv` and `sleep` data will automatically be used by `analyzeHealthMoodCorrelations.js` because they follow the unified schema.
- **Entry Context**: The `getEntryHealthContext` function will now automatically include Whoop metrics in journal entries.

---

5. UI Implementation

A. Settings Page (SettingsPage.jsx)

Add a new row for Whoop connectivity:

```javascript
{
  icon: Activity,
  label: 'Whoop Integration',
  description: isWhoopLinked ? 'Connected' : 'Link your Whoop account',
  onClick: onLinkWhoop,
}
```

B. Health Settings (HealthSettingsScreen.jsx)

Update the screen to display Whoop-specific status. Unlike HealthKit, which shows a "Use the Mobile App" message on web, Whoop should show a "Connect Cloud" button active in all environments.

---

6. Engineering Checklist

1. [ ] Register application in Whoop Developer Dashboard.
2. [ ] Add `WHOOP_CLIENT_SECRET` to Cloud Run environment variables.
3. [ ] Implement OAuth redirect handler in `relay-server/src/index.ts`.
4. [ ] Verify data normalization matches the schema in `src/services/health/healthKit.js` to ensure backward compatibility.
5. [ ] Test token refresh logic by manually expiring a session in Firestore.