Project Report: Google Calendar Sync Without WebSockets or BaaS

Project Title:

Real-Time Google Calendar Sync Using OAuth2, Push Notifications, and Smart Polling

Objective:

To create a real-time frontend UI that displays a user's Google Calendar events, responds to changes like additions/deletions, and remains in sync without using WebSockets or any Backend-as-a-Service (BaaS) platforms.

Technologies Used:

Project Structure:

```
google-calendar-sync/
├── backend/              → Express server for auth, webhooks
│   ├── server.js         → Main logic (OAuth, webhook, event
fetch)
│   └── .env              → Google API credentials & webhook
URL
├── frontend/             → React app to display calendar
events
│   └── src/App.jsx       → Smart polling UI to show events
```

Key Features:

OAuth2 Flow: Secure authentication using Google login.

Webhook Integration: Uses Google Calendar push notifications to detect changes.

Smart Polling: Polls only when the browser tab is visible to save API quota.

Minimal Backend: Self-hosted Express server without Firebase/Supabase/WebSockets.

Public Testing Tunnel: LocalTunnel used to expose webhook endpoint.

OAuth Configuration:

Client ID & Secret created via Google Cloud Console.

Redirect URI: `http://localhost:3001/auth/callback`

Webhook Endpoint: Public HTTPS tunnel (e.g., `https://tricky-mangos-march.loca.lt/notifications`)

Flow Summary:

1. User visits frontend → clicks **Login with Google**.
2. OAuth redirects to backend, exchanges code for tokens.
3. Backend registers a webhook (`watch`) for Google Calendar events.
4. On any calendar change, Google hits the webhook URL.
5. Backend fetches updated events and exposes them at `/events`.
6. Frontend polls `/events` every 5s (only if tab is visible) and updates UI.

🔍 Testing Outcome:

Successfully authenticated via Google OAuth.

Webhook was registered and received notifications.

Calendar events reflected on frontend with near real-time delay.

Entire setup worked without any WebSocket or 3rd-party backend service.

Known Limitations:

Webhook requires public HTTPS → used LocalTunnel (which resets URL on restart).

Tokens stored in-memory (not persistent across server restarts).

Google’s watch channel expires every 24 hours (needs periodic re-registration).