Product Requirements Document (PRD)

Unified Calendar Web Application

1. Executive Summary

1.1 Product Vision

The Unified Calendar Web Application aims to revolutionize how users manage their schedules by creating a seamless, centralized platform that integrates events from both Google and Microsoft calendars. Users will be able to view, create, edit, and manage all their events in one place without switching between applications.

1.2 Business Objectives

- Reduce scheduling fragmentation for users who maintain multiple calendars
- Increase productivity through streamlined calendar management
- Build a platform that can later expand to support additional calendar integrations
- Create a foundation for premium features through a freemium model

1.3 Success Metrics

- User adoption: 10,000 active users within 6 months
- Engagement: 70% of users connect both Google and Microsoft accounts
- Retention: 60% monthly active user retention
- User satisfaction: Net Promoter Score (NPS) of 40+

2. Product Overview

2.1 Product Description

The Unified Calendar is a web application that brings together events from Google Calendar and Microsoft Calendar into a single, intuitive interface. It enables bi-directional synchronization, allowing users to create and modify events in one place while ensuring changes propagate to the source calendars. The application includes smart scheduling assistance, unified notifications, and collaboration features across calendar platforms.

2.2 Target Audience

- Knowledge workers who use multiple calendar platforms
- Teams with mixed technology environments (Google Workspace and Microsoft 365)

- Project managers who coordinate across organizational boundaries
- Professionals who manage both work and personal calendars on different platforms

2.3 User Personas

Busy Professional (Primary)

Sarah, 34, Marketing Director

- Uses Google Calendar for personal events and Microsoft Outlook for work
- Struggles to avoid double-booking across calendars
- Needs a unified view of all commitments to plan effectively

Team Leader (Secondary)

Marcus, 42, Engineering Manager

- Manages teams using both Google and Microsoft ecosystems
- Needs to schedule meetings efficiently with team members across platforms
- Requires visibility into team availability regardless of calendar platform

Executive Assistant (Secondary)

Elena, 29, Executive Assistant

- Manages calendars for multiple executives
- Needs to quickly find available times across different calendar systems
- Requires reliable notification system to ensure executives don't miss meetings

3. Product Requirements

3.1 Technical Requirements

- Real-time synchronization between Google Calendar API and Microsoft Graph API
- Secure OAuth 2.0 authentication implementation
- Server-side token management and refresh
- Responsive web design supporting desktop and mobile browsers
- Low-latency data retrieval and rendering

3.2 Design Requirements

- Google Calendar-inspired UI for familiarity and ease of use

- Clear visual distinction between calendar sources
- Intuitive event creation and editing interface
- Consistent color-coding system for event sources
- Accessibility compliance with WCAG 2.1 AA standards

3.3 Development Stack

- **Frontend**: Next.js with TypeScript
- **Authentication**: NextAuth.js
- **Database**: Supabase
- **UI Framework**: Tailwind CSS with shadcn/ui components
- **Calendar Library**: FullCalendar

4. Feature Requirements

4.1 MVP 1: Unified Calendar - Read-Only View

User Story

As a user of the unified calendar web app, I want to connect my Google and Microsoft accounts and see all of my events in a single calendar view, so that I can get a clear picture of my schedule without switching between apps.

Functional Requirements

1. **Account Connection**
   - User creates an account with email/password or social login
   - User connects Google account via OAuth
   - User connects Microsoft account via OAuth
   - User can disconnect either account at will
2. **Calendar Display**
   - Events from both sources merged into unified view
   - Day, Week, and Month view options
   - Visual distinction between Google and Microsoft events (color-coding)
   - Events display accurate times, titles, and locations
   - Time zone support for correct event time display
3. **Calendar Navigation**

- Browse forward/backward in time
- Quickly jump to today
- Date selector for rapid navigation to specific dates

Non-Functional Requirements

- Events load within 2 seconds
- Calendar renders correctly across browsers (Chrome, Safari, Firefox, Edge)
- Secure token storage with proper encryption
- API request optimization to avoid rate limits

Acceptance Criteria

- User can successfully authenticate with both Google and Microsoft accounts
- All events from both sources appear in the calendar view
- Events clearly indicate their source through visual cues
- Calendar supports day/week/month view switching
- Calendar correctly shows events in user's time zone

4.2 MVP 2: Bi-Directional Sync & Event Management

User Story

As a user of the web app, I want to be able to create, edit, and delete events directly from the unified calendar, so that I can manage everything in one place and have it reflect in my Google or Microsoft calendar.

Functional Requirements

1. **Event Creation**
   - Create new events with title, location, time, description
   - Select destination calendar (Google or Microsoft)
   - Set event visibility (private/public)
   - Add video conferencing links
   - Set recurrence rules
2. **Event Editing**
   - Modify any event property
   - Update event time or duration

- Change event calendar source
- Save changes to the original calendar

3. **Event Deletion**
   - Delete single events
   - Delete recurring event series or individual occurrences
   - Synchronize deletions back to source calendar
4. **Sync Management**
   - Real-time updates when changes occur in source calendars
   - Conflict detection and resolution
   - Sync status indicators
   - Manual sync option

Acceptance Criteria

- New events created in the app appear in the selected source calendar
- Edited events reflect changes in both the app and source calendar
- Deleted events are removed from both the app and source calendar
- Changes made in Google/Microsoft calendars appear in the app within 2 minutes
- User receives notification when sync conflicts occur

4.3 MVP 3: Smart Availability & Scheduling Assistance

User Story

As a user scheduling meetings, I want to see my availability across both calendars in one place and get suggested meeting times, so that I can plan my meetings efficiently without double-booking.

Functional Requirements

1. **Availability View**
   - Combined free/busy times across both calendars
   - Visual indicators for busy, tentative, and free time slots
   - Toggle to show/hide specific calendars
2. **Meeting Suggestions**
   - Algorithm-generated optimal meeting time suggestions
   - Consideration of preferred working hours

- Respect for already scheduled events across calendars
- Duration-based suggestions (30 min, 1 hour, etc.)

3. **Attendee Management**
   - Add attendees from Google or Microsoft contacts
   - Check attendee availability if shared
   - Send invitations across platforms
4. **Time Zone Support**
   - Display availability in user's time zone
   - Convert times for attendees in different time zones
   - Indicate time zone differences in scheduling view

Acceptance Criteria

- Free/busy time correctly aggregated from both calendar platforms
- Visual indicators clearly show availability status
- Smart time suggestions avoid conflicts in all connected calendars
- Time zone conversions display correctly for international scheduling
- Attendee availability is incorporated into suggestions when available

4.4 MVP 4: Centralized Notifications & Reminders

User Story

As a user of the unified calendar, I want to get centralized notifications and reminders for my events, so that I never miss a meeting—regardless of where it was created.

Functional Requirements

1. **Notification Settings**
   - Global notification preferences
   - Per-calendar notification settings
   - Custom notification timing options
2. **Notification Delivery**
   - Email notifications
   - In-app notifications
   - Browser notifications (optional)

3. **Reminder Management**
    - Custom reminder timing (5 min, 10 min, 1 hour, 1 day before)
    - Multiple reminders per event
    - Snooze functionality
4. **Meeting Join Integration**
    - One-click access to video conferencing links
    - Support for Google Meet, Microsoft Teams, and Zoom links
    - Automatic detection of meeting links in event descriptions

Acceptance Criteria

- Notifications arrive at user-specified times
- Notification settings can be customized per calendar
- Meeting links are actionable directly from notifications
- Email notifications contain all relevant meeting details
- In-app notification center shows upcoming and missed events

4.5 MVP 5: Shared Calendars & Collaboration

User Story

As a team lead or project manager, I want to create and share unified calendars with my team, so that we can collaborate across platforms without worrying about who uses Google or Microsoft.

Functional Requirements

1. **Calendar Sharing**
    - Create shareable calendars within the app
    - Set permission levels (view/edit)
    - Generate sharing links
    - Send email invitations to collaborators
2. **Cross-Platform RSVP**
    - Attendees can respond from any calendar platform
    - RSVP status synchronized across platforms
    - Attendance tracking across calendar systems
3. **Team Calendars**

- Create dedicated team or project calendars
- Aggregate team member availability
- Find optimal meeting times for groups

4. **Collaboration Features**
   - Comments on events
   - Attachment support
   - Meeting agenda templates
   - Follow-up task creation

Acceptance Criteria

- Shared calendars accessible to all invited members
- Permission levels properly enforced
- RSVP status correctly synchronized between platforms
- Team availability accurately displayed for scheduling
- Collaborative features work seamlessly across Google and Microsoft accounts

5. User Interface Design

5.1 UI Components

Header Bar

- Application logo and name
- Today button
- Date navigation controls (prev/next)
- Current period display (Month/Year)
- View switcher (Day/Week/Month)
- Create event button
- User account menu

Sidebar

- My calendars section
  - Google calendars list with checkboxes
  - Microsoft calendars list with checkboxes
  - Color indicators for each calendar

- Other calendars section (shared)
- Settings and add calendar options

Main Calendar Area

- Time grid (for day/week views)
- Date grid (for month view)
- Event blocks with color-coding
- Visual indicators for multi-calendar events
- Current time indicator

Event Modal

- View mode with event details
- Edit mode with form controls
- Delete and update options
- Source calendar indicator
- Attendee management section

Create Event Modal

- Event title input
- Date and time selectors
- Location input
- Description editor
- Video conferencing options
- Calendar selector (Google vs Microsoft)
- Recurrence settings
- Reminder settings
- Save and cancel buttons

5.2 User Flows

Account Connection Flow

1. User signs up/logs in
2. User redirected to dashboard with connect prompts

3. User clicks "Connect Google Calendar"
4. OAuth consent screen appears
5. User grants permissions
6. Repeat for Microsoft Calendar
7. User sees unified calendar with events

Event Creation Flow

1. User clicks "Create" button
2. Create event modal appears
3. User fills out event details
4. User selects destination calendar
5. User clicks "Save"
6. Event appears in unified calendar and syncs to source

Calendar Sharing Flow

1. User navigates to calendar settings
2. User creates new shared calendar or selects existing
3. User adds team members with permissions
4. System sends invitations
5. Team members accept and see calendar
6. Calendar events sync bi-directionally

6. Technical Architecture

6.1 System Components

Authentication Layer

- NextAuth.js implementation
- OAuth 2.0 providers for Google and Microsoft
- JWT token management
- Refresh token handling

Data Layer

- Supabase database for user data and event caching

- Calendar connections table
- Events cache table
- User preferences table

API Layer

- Next.js API routes
- Google Calendar API integration
- Microsoft Graph API integration
- Error handling and rate limiting

UI Layer

- Next.js app router
- Tailwind CSS styling
- shadcn/ui components
- FullCalendar implementation

6.2 Data Models

User Model

```typescript
interface User {
  id: string;
  email: string;
  name: string;
  image?: string;
  created_at: Date;
  updated_at: Date;
}
```

CalendarConnection Model

typescript

```typescript
interface CalendarConnection {
  id: string;
  user_id: string;
  provider: 'google' | 'microsoft';
  provider_account_id: string;
  access_token: string;
  refresh_token?: string;
  expires_at: number;
  created_at: Date;
  updated_at: Date;
}
```

Event Model

typescript

```typescript
interface Event {
  id: string;
  user_id: string;
  provider: 'google' | 'microsoft' | 'local';
  provider_event_id: string;
  title: string;
  description?: string;
  start_time: Date;
  end_time: Date;
  location?: string;
  is_all_day: boolean;
  recurrence?: string;
  event_data: Record<string, any>;
  last_synced: Date;
}
```

CalendarSettings Model

```typescript
interface CalendarSettings {
  id: string;
  user_id: string;
  provider: 'google' | 'microsoft';
  calendar_id: string;
  name: string;
  color: string;
  visible: boolean;
  notifications_enabled: boolean;
}
```

6.3 API Endpoints

Authentication Endpoints

- `/api/auth/[...nextauth]` - NextAuth.js authentication routes
- `/api/auth/connect-google` - Connect Google Calendar
- `/api/auth/connect-microsoft` - Connect Microsoft Calendar

Calendar Endpoints

- `/api/calendars` - List user's calendars
- `/api/calendars/settings` - Update calendar settings

Event Endpoints

- `/api/events` - Get events (with filtering)
- `/api/events/create` - Create new event
- `/api/events/[id]` - Get, update, or delete event
- `/api/events/sync` - Trigger manual sync

Availability Endpoints

- `/api/availability` - Get free/busy information
- `/api/availability/suggest` - Get meeting time suggestions

Sharing Endpoints

- `/api/share/calendar` - Create shared calendar

- `/api/share/invite` - Invite users to calendar

7. Implementation Plan

7.1 Development Phases

Phase 1: Foundation (Weeks 1-2)

- Set up Next.js project with TypeScript
- Configure Supabase database
- Implement NextAuth.js authentication
- Create basic UI structure

Phase 2: MVP 1 - Unified Calendar View (Weeks 3-4)

- Implement Google and Microsoft OAuth flows
- Create calendar data fetching services
- Build unified calendar UI components
- Implement read-only calendar view

Phase 3: MVP 2 - Bi-Directional Sync (Weeks 5-6)

- Create event management interfaces
- Implement CRUD operations for events
- Build sync mechanisms between platforms
- Add conflict detection and resolution

Phase 4: MVP 3 - Smart Scheduling (Weeks 7-8)

- Develop availability calculation engine
- Create meeting suggestion algorithm
- Build scheduling interface components
- Implement time zone support

Phase 5: MVP 4 - Notifications (Weeks 9-10)

- Create notification system
- Implement delivery mechanisms
- Build notification preferences UI

- Add meeting join integrations

Phase 6: MVP 5 - Collaboration (Weeks 11-12)

- Develop calendar sharing functionality
- Create team calendar features
- Implement cross-platform RSVP
- Add collaboration tools

7.2 Testing Strategy

Unit Testing

- Jest for component and function testing
- React Testing Library for UI components
- 80% code coverage minimum

Integration Testing

- API endpoint testing
- Calendar synchronization testing
- Authentication flow testing

End-to-End Testing

- Cypress for user flow testing
- Cross-browser compatibility testing
- Mobile responsiveness testing

7.3 Deployment Strategy

- Development environment: Vercel Preview
- Staging environment: Vercel Preview (production configuration)
- Production environment: Vercel Production
- Database: Supabase Production Instance
- CI/CD: GitHub Actions

8. Post-Launch Considerations

8.1 Analytics and Monitoring

- Implement Google Analytics for user behavior tracking
- Set up error tracking with Sentry
- Create custom event tracking for key user actions
- Establish performance monitoring dashboards

8.2 Future Enhancements

- AI-powered scheduling assistant
- Mobile applications for iOS and Android
- Additional calendar integrations (Apple Calendar, Zoho, etc.)
- Calendar analytics and insights
- Team productivity features

8.3 Maintenance Plan

- Weekly dependency updates
- Bi-weekly bug fix releases
- Monthly feature updates
- Quarterly performance reviews

9. Appendix

9.1 Glossary

- **OAuth**: Open standard for access delegation
- **API**: Application Programming Interface
- **JWT**: JSON Web Token
- **Calendar Provider**: Source calendar system (Google or Microsoft)
- **Sync**: Synchronization between calendar systems

9.2 References

- Google Calendar API Documentation
- Microsoft Graph API Documentation
- NextAuth.js Documentation
- Supabase Documentation
- FullCalendar Documentation