



Mindful Champion Notification System

Overview

The Mindful Champion notification system is a comprehensive, database-driven notification infrastructure designed to keep users engaged with personalized, timely communications. Built with scalability and user experience in mind.



Architecture

Database Schema

1. NotificationPreferences

Stores user preferences for different notification categories.

```
model NotificationPreferences {
    id          String          @id @default(cuid())
    userId      String
    category    NotificationCategory // GOALS, VIDEO_ANALYSIS, TOURNAMENTS, etc
    .
    emailEnabled Boolean        @default(true)
    pushEnabled  Boolean        @default(true)
    inAppEnabled Boolean        @default(true)
    frequency   NotificationFrequency // DAILY, MULTIPLE, WEEKLY, CUSTOM
    customTimes Json?          // ["08:00", "12:00", "18:00"]
    timezone    String          @default("America/New_York")
    createdAt   DateTime        @default(now())
    updatedAt   DateTime        @updatedAt
}
```

Key Features:

- Per-category preferences
- Multiple delivery methods (Email, Push, In-App)
- Flexible frequency settings
- Timezone-aware scheduling
- Custom time slots

2. ScheduledNotification

Manages all scheduled notifications.

```

model ScheduledNotification {
    id          String           @id @default(cuid())
    userId      String
    category    NotificationCategory
    type        String           // goal_confirmation, daily_tip, etc.
    title       String
    message     String
    data        Json?            @db.Text
                                         // Additional payload
    scheduledFor DateTime
    status      NotificationStatus // PENDING, SENT, FAILED, CANCELLED
    deliveryMethod NotificationDeliveryMethod
    source      NotificationSource // SYSTEM, USER, COACH_KAI
    createdAt   DateTime         @default(now())
    sentAt     DateTime?
    history    NotificationHistory[]
}

```

Key Features:

- Flexible scheduling
- Rich data payload support
- Status tracking
- Source attribution
- Delivery method selection

3. NotificationHistory

Tracks delivery and engagement metrics.

```

model NotificationHistory [
    id          String           @id @default(cuid())
    userId      String
    notificationId String
    deliveredAt DateTime         @default(now())
    opened      Boolean          @default(false)
    clicked     Boolean          @default(false)
]

```

Key Features:

- Delivery confirmation
- Open tracking
- Click-through tracking
- Engagement analytics

4. CoachKaiReminder

Special reminders created through Coach Kai interactions.

```
model CoachKaiReminder {
    id          String      @id @default(cuid())
    userId      String
    reminderText String      @db.Text
    parsedData   Json?       // Structured data: time, frequency, etc.
    isActive     Boolean     @default(true)
    nextTrigger  DateTime?
    createdFrom  String      @default("chat")
    createdAt    DateTime    @default(now())
    updatedAt    DateTime    @updatedAt
}
```

Key Features:

- Natural language reminders
 - AI-parsed intent
 - Recurring support
 - Chat integration
-



Service Layer

notification-service.ts

Core notification functionality.

Key Functions:

`sendNotification(params)`

Send a notification immediately.

```
await sendNotification({
    userId: 'user_id',
    category: NotificationCategory.GOALS,
    type: 'goal_confirmation',
    title: '@ Goal Confirmed!',
    message: 'Your goal has been set!',
    data: { goalId: 'goal_123' },
});
```

`scheduleNotification(params)`

Schedule a notification for future delivery.

```
await scheduleNotification({
    userId: 'user_id',
    category: NotificationCategory.VIDEO_ANALYSIS,
    type: 'video_complete',
    title: '🎥 Analysis Ready!',
    message: 'Your video analysis is complete!',
    scheduledFor: new Date('2025-12-04T10:00:00Z'),
});
```

`cancelNotification(notificationId)`

Cancel a pending notification.

```
await cancelNotification('notification_id');
```

getUpcomingNotifications(userId, limit)

Fetch upcoming notifications for a user.

```
const upcoming = await getUpcomingNotifications('user_id', 10);
```

processScheduledNotifications()

Process all pending notifications (called by cron).

```
await processScheduledNotifications();
```

scheduling-service.ts

Handles recurring notifications and timezone-aware scheduling.

Key Functions:

scheduleRecurringNotification(params)

Schedule a recurring notification.

```
await scheduleRecurringNotification({
  userId: 'user_id',
  category: NotificationCategory.GOALS,
  type: 'daily_goal_tip',
  title: '🌟 Daily Tip',
  message: 'Your daily improvement tip!',
  frequency: NotificationFrequency.DAILY,
  timezone: 'America/New_York',
});
```

scheduleDailyGoalNotifications(userId, goalData)

Quick helper for daily goal notifications.

```
await scheduleDailyGoalNotifications('user_id', { goalId: 'goal_123' });
```

updateSchedule(userId, category, newSchedule)

Update notification schedule and cancel old notifications.

```
await updateSchedule('user_id', NotificationCategory.GOALS, {
  frequency: NotificationFrequency.MULTIPLE,
  customTimes: ['08:00', '12:00', '18:00'],
});
```

Email Templates

Located in `lib/notifications/email-templates.ts`

Available Templates:

1. Goal Confirmation (goal_confirmation)

- Confirms goal creation
- Outlines next steps
- Motivational tone

2. Daily Goal Tip (daily_goal_tip)

- Daily coaching tips
- Progress tracking
- Actionable advice

3. Video Analysis Complete (video_complete)

- Analysis ready notification
- Key insights preview
- Call to action

4. Trial Expiring (trial_expiring)

- Urgency messaging
- Feature highlights
- Special upgrade offer

5. Tournament Reminder (tournament_reminder)

- Event details
- Pre-tournament checklist
- Mental preparation tips

6. New Media Content (new_media)

- Content announcement
- Personalized recommendations
- Easy access link

7. Achievement Unlocked (achievement_unlocked)

- Celebration message
- Badge display
- Reward points info

Coach Kai Personality

All emails feature Coach Kai's signature style:

- Warm and encouraging
 - Expert but accessible
 - Action-oriented
 - Celebrates wins
 - Motivational quotes
 - Beautiful gradient designs
-



API Endpoints

Notification Preferences

GET /api/notifications/preferences

- Get all user notification preferences
- Returns array of preferences by category

PUT /api/notifications/preferences

- Update preferences for a category
- Body: { category, emailEnabled, pushEnabled, inAppEnabled, frequency, customTimes, timezone }

Scheduled Notifications

GET /api/notifications/scheduled

- Get user's scheduled notifications
- Query params: ?status=PENDING&category=GOALS&limit=50

POST /api/notifications/scheduled

- Create a scheduled notification
- Body: { category, type, title, message, scheduledFor, data, deliveryMethod }

DELETE /api/notifications/scheduled/[id]

- Cancel a scheduled notification

Notification History

GET /api/notifications/history

- Get notification history
- Query params: ?limit=50&offset=0
- Returns history with notification details

PATCH /api/notifications/history/[id]

- Update history record (mark as opened(clicked))
- Body: { opened: true, clicked: true }

Coach Kai Reminders

GET /api/notifications/reminders

- Get user's reminders
- Query params: ?isActive=true

POST /api/notifications/reminders

- Create a new reminder
- Body: { reminderText, parsedData, nextTrigger, createdFrom }

PUT /api/notifications/reminders/[id]

- Update a reminder
- Body: { reminderText, parsedData, nextTrigger, isActive }

DELETE /api/notifications/reminders/[id]

- Delete a reminder

Cron Jobs

GET /api/cron/process-notifications

- Process all pending notifications

- Requires `Authorization: Bearer <CRON_SECRET>` header
 - Should be called every 5-15 minutes
-

Setup & Configuration

1. Environment Variables

Add to `.env` :

```
# Notification System
CRON_SECRET=your-secure-random-string-here
NEXT_PUBLIC_APP_URL=https://your-app-url.com

# Email Service (choose one)
# SendGrid
SENDGRID_API_KEY=your-sendgrid-key

# AWS SES
AWS_ACCESS_KEY_ID=your-aws-key
AWS_SECRET_ACCESS_KEY=your-aws-secret
AWS_REGION=us-east-1

# Push Notifications (optional)
FIREBASE_SERVER_KEY=your-firebase-key
```

2. Database Migration

Already completed! Schema is in production.

```
# If needed to regenerate
npx prisma generate
npx prisma db push
```

3. Set Up Cron Job

Vercel:

Create `vercel.json` :

```
{
  "crons": [
    {
      "path": "/api/cron/process-notifications",
      "schedule": "*/10 * * * *"
    }
  ]
}
```

AWS EventBridge:

```
# Create a rule that triggers every 10 minutes
aws events put-rule --schedule-expression "rate(10 minutes)" --name process-notifications

# Add target
aws events put-targets --rule process-notifications --targets "Id"="1","Arn"="your-lambda-arn"
```

4. Email Service Integration

Update `lib/notifications/email-templates.ts`:

For SendGrid:

```
import sgMail from '@sendgrid/mail';

sgMail.setApiKey(process.env.SENDGRID_API_KEY);

export async function sendNotificationEmail(params: EmailTemplateParams) {
  // ... existing code ...

  await sgMail.send({
    to: user.email,
    from: 'coach@mindfulchampion.com',
    subject: emailContent.subject,
    html: emailContent.html,
  });
}
```

For AWS SES:

```
import { SESClient, SendEmailCommand } from '@aws-sdk/client-ses';

const ses = new SESClient({ region: process.env.AWS_REGION });

export async function sendNotificationEmail(params: EmailTemplateParams) {
  // ... existing code ...

  await ses.send(new SendEmailCommand({
    Source: 'coach@mindfulchampion.com',
    Destination: { ToAddresses: [user.email] },
    Message: {
      Subject: { Data: emailContent.subject },
      Body: { Html: { Data: emailContent.html } },
    },
  }));
}
```

🧪 Testing

Run the test suite:

```
npx tsx scripts/tests/test-notification-system.ts
```

Test results show:

- Notification preferences
- Schedule notifications
- Upcoming notifications
- Immediate notifications
- Recurring notifications
- Coach Kai reminders
- Notification history
- Cancel notifications



Usage Examples

Example 1: Send Goal Confirmation

```
import { sendNotification } from '@lib/notifications/notification-service';
import { NotificationCategory } from '@prisma/client';

await sendNotification({
  userId: user.id,
  category: NotificationCategory.GOALS,
  type: 'goal_confirmation',
  title: '⌚ Your Goal is Set – Let's Make It Happen!',
  message: 'Your goal has been confirmed!',
  data: {
    goalType: 'Skill Improvement',
    goalDescription: 'Improve backhand consistency',
    targetDate: '2025-12-31',
  },
});
```

Example 2: Schedule Weekly Tournament Reminder

```
import { scheduleNotification } from '@lib/notifications/notification-service';
import { NotificationCategory } from '@prisma/client';

const tournamentDate = new Date('2025-12-15T09:00:00Z');
const reminderDate = new Date(tournamentDate);
reminderDate.setDate(reminderDate.getDate() - 7); // 7 days before

await scheduleNotification({
  userId: user.id,
  category: NotificationCategory.TOURNAMENTS,
  type: 'tournamentReminder',
  title: '🏆 Tournament Coming Up!',
  message: 'Your tournament is in 7 days!',
  scheduledFor: reminderDate,
  data: {
    tournamentName: 'City Championship',
    tournamentDate: '2025-12-15',
    location: 'Main Street Courts',
    daysUntil: 7,
  },
});
```

Example 3: Set Up User Preferences

```
import { prisma } from '@/lib/db';
import { NotificationCategory, NotificationFrequency } from '@prisma/client';

await prisma.notificationPreferences.create({
  data: {
    userId: user.id,
    category: NotificationCategory.GOALS,
    emailEnabled: true,
    pushEnabled: true,
    inAppEnabled: true,
    frequency: NotificationFrequency.MULTIPLE,
    customTimes: ['08:00', '12:00', '18:00'],
    timezone: 'America/Los_Angeles',
  },
});
```

Example 4: Create Coach Kai Reminder from Chat

```
import { prisma } from '@/lib/db';

// When Coach Kai processes: "Remind me to practice every morning at 8 AM"
await prisma.coachKaiReminder.create({
  data: {
    userId: user.id,
    reminderText: 'Remind me to practice every morning at 8 AM',
    parsedData: {
      task: 'practice',
      frequency: 'daily',
      time: '08:00',
    },
    nextTrigger: calculateNextMorning8AM(user.timezone),
    isActive: true,
    createdFrom: 'chat',
  },
});
```

🎯 Default Notification Settings

When a user signs up, set default preferences:

```

const defaultCategories = [
  NotificationCategory.GOALS,
  NotificationCategory.VIDEO_ANALYSIS,
  NotificationCategory.TOURNAMENTS,
  NotificationCategory.ACHIEVEMENTS,
  NotificationCategory.COACH_KAI,
];

for (const category of defaultCategories) {
  await prisma.notificationPreferences.create({
    data: {
      userId: newUser.id,
      category,
      emailEnabled: true,
      pushEnabled: true,
      inAppEnabled: true,
      frequency: NotificationFrequency.DAILY,
      timezone: newUser.timezone || 'America/New_York',
    },
  });
}

```

Analytics & Monitoring

Track Key Metrics:

1. Delivery Rate

```

const totalSent = await prisma.scheduledNotification.count({
  where: { status: 'SENT' }
});

const totalFailed = await prisma.scheduledNotification.count({
  where: { status: 'FAILED' }
});

const deliveryRate = (totalSent / (totalSent + totalFailed)) * 100;

```

1. Open Rate

```

const opened = await prisma.notificationHistory.count({
  where: { opened: true }
});

const total = await prisma.notificationHistory.count();
const openRate = (opened / total) * 100;

```

1. Click-Through Rate

```
const clicked = await prisma.notificationHistory.count({
  where: { clicked: true }
});

const total = await prisma.notificationHistory.count();
const ctr = (clicked / total) * 100;
```

Security Considerations

1. Cron Job Authentication

- Always use `CRON_SECRET` for cron endpoints
- Rotate secret regularly

2. User Authorization

- All API routes check user session
- Users can only access their own notifications

3. Rate Limiting

- Implement rate limits on notification creation
- Prevent spam/abuse

4. Data Privacy

- Notification data follows GDPR compliance
- Users can delete their notification history

Troubleshooting

Notifications Not Sending

1. Check cron job is running
2. Verify email service credentials
3. Check notification status in database
4. Review logs for errors

Wrong Timezone

1. Verify user timezone in preferences
2. Check `getTimezoneOffset()` mapping
3. Use proper timezone library (`moment-timezone`, `date-fns-tz`)

Duplicate Notifications

1. Check for duplicate cron job triggers
2. Add idempotency checks
3. Use unique constraints where appropriate



User Interface Components

Notification Preferences Page

Located at `/settings/notifications`, provides a comprehensive UI for managing notification preferences.

Components:

1. NotificationPreferences (`components/notifications/notification-preferences.tsx`)

Main container component that:

- Fetches all notification preferences
- Manages state for unsaved changes
- Handles save/cancel operations
- Displays success/error messages
- Shows all notification categories

Features:

- Real-time change tracking
- Optimistic UI updates
- Error handling with user feedback
- Automatic preference loading
- Toast notifications for actions

2. CategoryPreference (`components/notifications/category-preferences.tsx`)

Individual category card with:

- Expandable/collapsible sections
- Quick enable/disable toggle
- Delivery method checkboxes (Email, Push, In-App)
- Frequency selector (Daily, Multiple, Weekly, Custom)
- Custom time picker
- Timezone selector
- Beautiful gradient headers per category

Categories:

-  **Goals & Training** - Progress updates, milestones, training reminders
-  **Video Analysis** - Analysis complete, insights ready, improvement suggestions
-  **Tournament Hub** - Live matches, tournament updates, bracket changes
-  **Media Center** - New content, live streams, podcast episodes
-  **Account & Subscription** - Billing, trial status, account updates
-  **Achievements** - New badges, tier unlocks, milestones reached
-  **Coach Kai Reminders** - Training tips, practice reminders, coaching insights

3. TimePicker (`components/notifications/time-picker.tsx`)

Custom time selection component:

- Add multiple notification times
- 12-hour format with AM/PM
- 15-minute intervals
- Visual time chips
- Remove times individually
- Automatic sorting

Example:

```
<TimePicker
  times={['08:00', '12:00', '18:00']}
  onChange={(newTimes) => handleTimesChange(newTimes)}
/>
```

4. QuickToggle (components/notifications/quick-toggle.tsx)

Inline notification toggle for feature pages:

- Quick enable/disable for specific categories
- Displayed on relevant pages (Goals, Video Analysis, Media, etc.)
- Links to full preferences page
- Updates preferences in real-time
- Toast feedback

Example Usage:

```
<QuickToggle
  category="GOALS"
  title="Goals & Training Notifications"
  description="Get updates on your progress, milestones, and training reminders"
/>
```

Integrated On:

- /progress/goals - Goals page
- /train/video - Video Analysis Hub
- /media - Tournament/Media Hub
- Feature-specific pages

Navigation

The notification preferences are accessible from:

1. Settings Page (/settings)

- “Notifications” card with bell icon
- Direct link to /settings/notifications

1. Feature Pages - Quick toggle cards with “Customize preferences” link

2. User Menu - Future: notification bell icon in header

UI/UX Features**Design Principles:**

- Clean, modern design with Tailwind CSS
- Gradient accents matching Mindful Champion brand
- Responsive on all devices
- Smooth animations with Framer Motion
- Accessible with proper ARIA labels
- Clear visual feedback for all actions

Interaction Patterns:

- Expand/collapse for detailed settings
- Master toggle to disable all methods for a category

- Individual toggles for fine-grained control
- Save/Cancel buttons with change tracking
- Inline validation and error messages

Visual Elements:

- Category-specific gradient headers
- Icon indicators for each category
- Badge chips for selected times
- Color-coded feedback (green for success, red for errors, amber for warnings)
- Loading states with spinners

API Integration

The UI components integrate seamlessly with the API:

Fetching Preferences:

```
const response = await fetch('/api/notifications/preferences');
const { preferences } = await response.json();
```

Updating Preferences:

```
await fetch('/api/notifications/preferences', {
  method: 'PUT',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({ preferences: updatedPreferences })
});
```

Quick Toggle:

```
// Fetches current state, updates specific category, saves back
const handleToggle = async (checked: boolean) => {
  const getResponse = await fetch('/api/notifications/preferences');
  const data = await getResponse.json();

  const updatedPreferences = data.preferences.map((pref: any) => {
    if (pref.category === category) {
      return { ...pref, emailEnabled: checked, pushEnabled: checked, inAppEnabled: checked };
    }
    return pref;
  });

  await fetch('/api/notifications/preferences', {
    method: 'PUT',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ preferences: updatedPreferences })
  });
};
```

Reminders Dashboard

Overview

The Reminders Dashboard is a comprehensive user interface for managing all reminders and viewing notification activity. Located at `/dashboard/reminders`, it provides a centralized hub for users to create, edit, delete, and monitor their notification reminders.

Features

1. Main Dashboard (`app/dashboard/reminders/page.tsx`)

Summary Cards:

- **Active Reminders** - Total count of active scheduled notifications
- **Next Notification** - Time until the next scheduled notification
- **This Week** - Total notifications sent in the past 7 days
- **Open Rate** - Percentage of opened notifications

Quick Actions:

- **Set Goal Reminder** - Quick setup for morning motivation or evening reflection
- **Custom Reminder** - Create a fully customized reminder
- **Manage Preferences** - Link to notification settings

Tabs:

- **Active** - View and manage all active reminders
- **Upcoming** - Timeline view of next 7 days of scheduled notifications
- **History** - View past notifications with filters and analytics

2. Components

RemindersDashboard (`components/notifications/reminders-dashboard.tsx`)

Main orchestrator component that:

- Fetches and displays all reminders
- Manages tabs and active view state
- Coordinates quick actions
- Real-time statistics updates
- Responsive layout for mobile/desktop

ReminderCard (`components/notifications/reminder-card.tsx`)

Individual reminder display with:

- Title, message, and category display
- Source badge (User, Coach Kai, System)
- Next trigger time with relative formatting
- Enable/disable toggle
- Edit and delete actions
- Status indicators (Active/Paused)

Features:

- Real-time status updates
- Inline editing capability
- Confirmation dialogs for destructive actions
- Visual indicators for different reminder types
- Category-specific icons and colors

UpcomingNotifications (components/notifications/upcoming-notifications.tsx)

Timeline view component with:

- Grouped by date (Today, Tomorrow, specific dates)
- Time-ordered display
- Category badges and colors
- Auto-refresh every minute
- Empty state for no upcoming notifications

Visual Design:

- Timeline-style layout
- Date headers with gradient backgrounds
- Individual notification cards with hover effects
- Time badges for scheduled delivery
- Category and delivery method indicators

NotificationHistory (components/notifications/notification-history.tsx)

Historical view with:

- Paginated list of sent notifications
- Filters by category and date range
- Analytics summary (total sent, open rate, click rate)
- Open/click status indicators
- Date range selector (7, 30, 90 days)

Analytics Metrics:

- Total Sent - Count of notifications delivered
- Open Rate - Percentage of notifications opened
- Click Rate - Percentage of notifications clicked
- Engagement Rate - Combined metric

Filters:

- Category dropdown (All, Goals, Video Analysis, etc.)
- Date range selector
- Pagination controls

QuickSetupModal (components/notifications/quick-setup-modal.tsx)

Pre-configured reminder templates:

Preset Options:

1. Morning Motivation (8:00 AM)

- Start your day with your goals
- Daily frequency
- Motivational messaging

1. Evening Reflection (8:00 PM)

- Reflect on daily progress
- Daily frequency
- Reflective messaging

2. Midday Check-in (12:00 PM)

- Quick progress check at noon
- Daily frequency
- Reminder messaging

Features:

- Visual card selection
- One-click setup
- Automatic scheduling (if time passed, schedules for next day)
- Beautiful gradient icons
- Preview of reminder message

CreateReminderForm (components/notifications/create-reminder-form.tsx)

Full custom reminder creation with:

- Title input (required)
- Message textarea (optional)
- Category selector
- Date picker (minimum: today)
- Time picker
- Frequency selector (Once, Daily, Weekly, Custom)
- Delivery method (In-App, Email, Both)

Validation:

- Required fields highlighted
- Date cannot be in the past
- Real-time form validation
- Clear error messages

EditReminderModal (components/notifications/edit-reminder-modal.tsx)

Edit existing reminders:

- Pre-populated form with current values
- Same fields as create form
- Updates scheduled notification
- Maintains notification history

API Integration

Reminders Management

GET /api/notifications/reminders

- Fetch all user reminders (scheduled + Coach Kai)
- Query params: ?status=active
- Returns combined list with totals

POST /api/notifications/reminders

- Create new scheduled reminder
- Body: { category, title, message, scheduledFor, frequency, deliveryMethod, customTimes }
- Validates ownership and data
- Returns created reminder

PUT /api/notifications/reminders

- Update existing reminder
- Body: { id, ...updateData }
- Verifies user ownership
- Returns updated reminder

DELETE /api/notifications/reminders?id=xxx

- Delete a reminder

- Verifies ownership
- Returns success status

Scheduled Notifications

GET /api/notifications/scheduled?days=7

- Fetch upcoming notifications for next N days
- Returns notifications grouped by date
- Used for timeline view

History & Analytics

GET /api/notifications/history?dateRange=30&category=G0ALS&page=1

- Fetch notification history with pagination
- Filters by category and date range
- Returns analytics summary:
- Total sent, opened, clicked
- Open rate, click rate percentages
- Pagination metadata

Navigation Integration

The Reminders Dashboard is accessible from:

1. User Profile Dropdown (Desktop)

- Bell icon with "Reminders" label
- Located after "Connect Devices" option
- Tooltip: "Manage your reminders and notification settings"

2. Mobile Navigation Menu

- "Reminders" button in Account section
- Bell icon indicator
- Same placement as desktop

3. Direct Link - </dashboard/reminders>

Design System

Brand Colors:

- Teal gradient (from-teal-600 to-cyan-600) for primary actions
- Category-specific colors:
- Goals: Teal
- Video Analysis: Cyan
- Tournaments: Purple
- Media: Blue
- Achievements: Amber
- Coach Kai: Pink

Visual Elements:

- Summary cards with gradient borders
- Smooth animations with Framer Motion
- Responsive grid layouts
- Empty states with helpful illustrations
- Loading skeletons for better UX
- Toast notifications for feedback

Icons:

- Bell for reminders/notifications
- Clock for timing
- Calendar for scheduling
- Target for goals
- Sparkles for special features
- Trending Up for analytics

User Experience Features

Real-time Updates:

- Auto-refresh upcoming notifications every minute
- Optimistic UI updates for toggles
- Instant feedback for actions
- Loading states prevent duplicate actions

Mobile Optimization:

- Responsive layouts for all screen sizes
- Touch-friendly tap targets
- Swipe-friendly card interactions
- Collapsible sections on mobile
- Bottom sheets for modals

Accessibility:

- ARIA labels for all interactive elements
- Keyboard navigation support
- Screen reader friendly
- High contrast ratios
- Focus indicators

Empty States:

- Helpful illustrations
- Clear call-to-action buttons
- Guidance for getting started
- Encouragement to create first reminder

Usage Examples

Quick Setup Flow

```
// User clicks "Set Goal Reminder" in Quick Actions
// -> QuickSetupModal opens with 3 preset options
// -> User selects "Morning Motivation"
// -> Reminder created automatically at 8:00 AM daily
// -> User sees success toast
// -> Dashboard refreshes with new reminder
```

Custom Reminder Creation

```
// User clicks "New Reminder" button
// -> CreateReminderForm modal opens
// -> User fills: "Practice Drills" @ 6:00 PM today, Daily
// -> Submits form
// -> API creates scheduled notification
// -> Modal closes, dashboard refreshes
// -> New reminder appears in Active tab
```

Edit Existing Reminder

```
// User clicks edit icon on a reminder card
// -> EditReminderModal opens with pre-filled data
// -> User changes time from 8:00 AM to 9:00 AM
// -> Saves changes
// -> API updates scheduled notification
// -> Card updates in real-time
```

View Upcoming Timeline

```
// User switches to "Upcoming" tab
// -> Fetches next 7 days of notifications
// -> Groups by date (Today, Tomorrow, etc.)
// -> Shows timeline with times and details
// -> Auto-refreshes every minute
```

Review History & Analytics

```
// User switches to "History" tab
// -> Shows analytics cards (sent, open rate, etc.)
// -> Lists past notifications with pagination
// -> User filters by "Goals" category
// -> User selects "Last 7 days" date range
// -> List updates with filtered results
```

Technical Implementation

State Management:

- React hooks for local state
- API calls with fetch
- Real-time updates with intervals
- Optimistic UI updates

Data Flow:

```
User Action
  → Component Handler
  → API Call (fetch)
  → Backend Route
  → Prisma Database Query
  → Response
  → Component Update
  → UI Refresh
```

Error Handling:

- Try-catch blocks in all API calls
- User-friendly error messages
- Toast notifications for feedback
- Graceful degradation
- Retry mechanisms

Performance:

- Lazy loading for modals
- Pagination for history
- Debounced API calls
- Memoized components
- Efficient re-renders

Future Enhancements for Reminders Dashboard

- [] Bulk actions (delete multiple, enable/disable all)
- [] Drag-and-drop to reorder reminders
- [] Reminder templates library
- [] Share reminders with friends/partners
- [] Voice-based reminder creation
- [] Calendar integration (Google, Apple, Outlook)
- [] Snooze functionality
- [] Reminder categories organization
- [] Export reminders to CSV/JSON
- [] Recurring reminder advanced rules (every 2 weeks, monthly, etc.)
- [] Notification sound preview
- [] Dark mode optimization
- [] Keyboard shortcuts
- [] Search and filter in active reminders
- [] Reminder statistics and trends
- [] Integration with Coach Kai chat (create reminders via chat)
- [] Location-based reminders (when at court, etc.)
- [] Smart suggestions based on usage patterns

Goal-Based Notification System

Overview

The goal-based notification system provides comprehensive support for user goals with personalized daily check-ins, milestone celebrations, and actionable tips from Coach Kai. The system automatically manages the full goal lifecycle from creation to completion.

Features

1. Immediate Goal Confirmation Email

When a user creates a goal, they receive an immediate confirmation email containing:

- Goal details and target date
- What to expect (daily check-ins, progress tracking, milestone celebrations)

- Direct link to view and manage the goal
- Coach Kai's motivational message

2. Daily Progress Check-ins

Users receive daily emails (at their preferred time) with:

- Progress statistics (days into goal, days until target)
- Daily tip from Coach Kai (rotates through 5 categories)
- Motivational quote
- Direct link to log progress
- Tracking and accountability support

3. Milestone Achievement Notifications

When users complete milestones, they receive celebration emails with:

- Milestone achievement confirmation
- Progress visualization
- Coach Kai's congratulatory message
- Next steps and encouragement

Architecture

Core Services

1. Goal Tip Generator (`lib/notifications/goal-tip-generator.ts`)

Generates personalized tips across 5 categories:

Category	Focus	Tips Available
Technique	Paddle grip, stance, form	5 tips
Mental	Visualization, mindset, self-talk	5 tips
Practice	Drills, structure, consistency	5 tips
Tracking	Progress monitoring, analytics	5 tips
Motivation	Inspiration, perseverance	5 tips

Tip Rotation Logic:

- Tips rotate based on goal category and day number
- Ensures variety and prevents repetition
- Category focus aligns with goal type
- Each tip includes title, content, actionable step, and icon

Example Tip Structure:

```
{
  category: 'technique',
  title: "Perfect Your Paddle Grip",
  content: "A proper continental grip gives you control and power...",
  actionable: "Practice holding the paddle correctly for 5 minutes...",
  icon: "🎯"
}
```

2. Goal Notifications Service (lib/notifications/goal-notifications.ts)

Main notification orchestration:

Functions:

sendGoalConfirmation(userId, goalData)

Sends immediate goal confirmation email.

```
await sendGoalConfirmation(userId, {
  id: goal.id,
  title: "Master the Third Shot Drop",
  description: "Improve consistency from 40% to 80%",
  category: "SKILL_IMPROVEMENT",
  targetDate: new Date("2025-12-31"),
  userId: userId,
  progress: 0,
  createdAt: new Date()
});
```

setupDailyGoalReminders(userId, goalData, preferences)

Creates scheduled daily notifications with preferences:

```
await setupDailyGoalReminders(userId, goalData, {
  enableDailyReminders: true,
  preferredTime: "08:00",
  frequency: "daily" // or "every_other_day", "weekly"
});
```

sendDailyGoalProgressEmail(userId, goalId, dayNumber)

Sends daily progress email with tip:

```
await sendDailyGoalProgressEmail(userId, goalId, 5);
// Day 5 receives technique tip #2, mental tip on day 10, etc.
```

sendMilestoneAchievedEmail(userId, goalId, milestoneTitle)

Celebrates milestone completion:

```
await sendMilestoneAchievedEmail(
  userId,
  goalId,
  "Complete 50 successful third shot drops"
);
```

cancelGoalReminders(goalId)

Cancels all reminders when goal is completed or deleted:

```
await cancelGoalReminders(goalId);
// Automatically called when goal status changes to COMPLETED/ARCHIVED
```

Email Templates

1. Goal Confirmation Email

Subject: “🎯 Goal Set! Coach Kai is Here to Help”

Content:

- Welcome header with goal icon
- Personalized greeting
- Goal details card (title, description, category, target date)
- What's Next section (3-step process)
- Coach Kai motivational quote
- CTA button to view goal
- Link to notification settings

Design:

- Gradient backgrounds (teal to cyan)
- White content cards with shadows
- Numbered steps with gradient circles
- Mobile-responsive layout
- Coach Kai branding throughout

2. Daily Progress Check-in Email

Subject: “🌟 Daily Check-in: [Goal Title]”

Content:

- Good morning greeting
- Progress stats cards (days into goal, days until target)
- Today's tip section with:
 - Category badge (technique, mental, practice, etc.)
 - Tip title and content
 - Action item in highlighted box
 - Motivational quote in gradient card
 - CTA button to log progress
- Coach Kai signature

Dynamic Elements:

- Tips rotate based on day number
- Progress stats update automatically
- Motivational quotes cycle through 15 options
- Category-specific tips based on goal type

3. Milestone Achievement Email

Subject: “🏆 Milestone Reached: [Milestone Title]”

Content:

- Celebration header with trophy animation
- Congratulations message
- Achievement card with milestone details
- Coach Kai's proud message

- What's next list (celebrate, review next milestone, keep going)
- CTA button to view progress
- Encouraging footer

Design:

- Gold/orange gradients for celebration theme
- Large trophy icon
- Emphasizes achievement and progress
- Encourages continued effort

User Interface Integration

1. Goal Creation Dialog Enhancement

Added notification preferences section in create goal dialog:

Features:

- Toggle switch for daily check-ins
- Time picker for preferred notification time
- Frequency selector (daily, every other day, weekly)
- Pro tip callout about 42% increase in goal achievement
- Collapsible section with smooth animations

Code Example:

```
<Switch
  checked={notificationPreferences.enableDailyReminders}
  onCheckedChange={(checked) =>
    setNotificationPreferences({
      ...notificationPreferences,
      enableDailyReminders: checked
    })
  }
/>

{notificationPreferences.enableDailyReminders && (
  <motion.div>
    <Input type="time" value={notificationPreferences.preferredTime} />
    <Select value={notificationPreferences.frequency}>
      <SelectItem value="daily">Every Day</SelectItem>
      <SelectItem value="every_other_day">Every Other Day</SelectItem>
      <SelectItem value="weekly">Once a Week</SelectItem>
    </Select>
  </motion.div>
)}
```

2. Notification Preferences Integration

Goal notifications integrate with existing notification preferences system:

- Listed under “Goals & Training” category (⌚)
- User can adjust settings in /settings/notifications
- Preferences include email, push, and in-app options
- All managed through unified preferences UI

API Integration

Goal Creation API (/api/goals)

POST /api/goals

Enhanced to handle notification preferences:

```
{
  title: "Master the Third Shot Drop",
  description: "Improve consistency",
  category: "SKILL_IMPROVEMENT",
  targetDate: "2025-12-31",
  milestones: [
    { title: "Learn proper technique", description: "..." },
    { title: "Practice 100 reps", description: "..." }
  ],
  notificationPreferences: {
    enableDailyReminders: true,
    preferredTime: "08:00",
    frequency: "daily"
  }
}
```

Response Flow:

1. Creates goal in database
2. Sends immediate confirmation email
3. Sets up daily reminder schedule (if enabled)
4. Returns created goal with milestones

Milestone Management APIs

PATCH /api/goals/[goalId]/milestones/[milestoneId]

Updates milestone and triggers achievement notifications:

```
{
  status: "COMPLETED", // Triggers achievement email
  currentValue: 100,
  title: "Milestone title",
  description: "Milestone description"
}
```

Auto-updates:

- Sends achievement email when status changes to COMPLETED
- Updates parent goal progress percentage
- Maintains milestone completion history

Notification Processing Worker

Goal Notification Processor (lib/workers/process-goal-notifications.ts)

Purpose: Processes scheduled goal notifications

Main Function: processGoalDailyCheckIns()

Logic:

1. Finds all pending GOAL_DAILY_CHECKIN notifications due now

2. Verifies goal still exists and is active
3. Calculates day number for tip rotation
4. Sends daily progress email with appropriate tip
5. Marks notification as SENT
6. Schedules next notification based on frequency
7. Stops scheduling if goal completed or target date passed

Example Scheduling:

```
// Day 1: Technique tip #1
// Day 2: Mental tip #1
// Day 3: Practice tip #1
// Day 4: Technique tip #2
// ... continues rotating through categories
```

Helper Function: `checkOverdueGoals()`

- Identifies goals past target date but not completed
- Can be used for overdue reminders (future enhancement)

Cron Job Setup

Endpoint: `/api/cron/process-goal-notifications`

Purpose: Process all pending goal notifications

Authentication: Requires Bearer token matching `CRON_SECRET`

Frequency: Should run every 5-15 minutes for timely delivery

Response:

```
{
  "success": true,
  "timestamp": "2025-12-03T10:00:00Z",
  "checkIns": {
    "success": true,
    "processed": 5
  },
  "overdueGoals": {
    "success": true,
    "overdueCount": 2
  }
}
```

Setup Examples:

Vercel Cron:

```
{
  "crons": [
    {
      "path": "/api/cron/process-goal-notifications",
      "schedule": "*/10 * * * *"
    }
  ]
}
```

Manual Trigger (Development):

```
curl -X POST http://localhost:3000/api/cron/process-goal-notifications \
-H "Authorization: Bearer dev-secret-key"
```

Goal Notification Preferences

Users can customize goal notifications through three levels:

1. Goal Creation Time

- Enable/disable daily reminders
- Set preferred time (e.g., 08:00)
- Choose frequency (daily, every other day, weekly)

2. Notification Category Settings

- Manage all goal notifications in preferences UI
- Toggle email, push, and in-app delivery
- Set custom notification times
- Adjust frequency

3. Individual Reminder Management

- View all active goal reminders in dashboard
- Edit specific reminder times
- Disable individual reminders
- View upcoming schedule

Goal Progress Tracking Integration

Automatic Progress Updates

When milestones are completed:

```
// Calculate new progress
const completedCount = goal.milestones.filter(m => m.status === 'COMPLETED').length;
const totalCount = goal.milestones.length;
const newProgress = (completedCount / totalCount) * 100;

// Update goal
await prisma.goal.update({
  where: { id: goalId },
  data: { progress: newProgress }
});
```

Progress Visibility

Daily emails show:

- Days into goal (e.g., “Day 5”)
- Days until target (e.g., “25 days left”)
- Current progress percentage
- Recent milestone completions

Coach Kai Personality Integration

All goal notification emails feature Coach Kai’s signature style:

Tone:

- Warm and encouraging
- Expert but approachable
- Action-oriented
- Celebrates progress
- Maintains accountability

Visual Branding:

- Teal/cyan gradients (Coach Kai colors)
- Robot emoji  in signature
- Energetic emojis throughout
- Modern, clean design
- Mobile-optimized layouts

Language Patterns:

- "Hey champ!" greetings
- Motivational quotes
- Actionable advice
- Specific, concrete steps
- Progress celebration

Testing

Manual Testing Flow

1. Create Goal:

- Navigate to Goals page
- Click "Create Goal"
- Fill in goal details
- Enable daily reminders
- Set preferred time and frequency
- Submit

2. Verify Confirmation:

- Check email for confirmation
- Verify goal details displayed correctly
- Check "What's Next" section
- Click CTA to view goal

3. Check Scheduled Notifications:

- Query database: `prisma.scheduledNotification.findMany()`
- Verify next notification scheduled correctly
- Check frequency and time match preferences

4. Trigger Daily Check-in:

- Run cron manually or wait for scheduled time
- Verify email received
- Check tip content and category
- Verify progress stats accurate

5. Complete Milestone:

- Mark milestone as completed in UI
- Check achievement email received

- Verify goal progress updated
- Confirm notification contains correct milestone

6. Manage Preferences:

- Navigate to notification settings
- Update goal notification preferences
- Verify changes reflected in scheduled notifications

Automated Testing

```
# Test tip generation
node -e "
  const { generateGoalTip } = require('./lib/notifications/goal-tip-generator');
  for (let i = 0; i < 10; i++) {
    const tip = generateGoalTip({
      title: 'Test Goal',
      category: 'SKILL_IMPROVEMENT'
    }, i);
    console.log(`Day ${i}: ${tip.category} - ${tip.title}`);
  }
"

# Test notification processing
curl -X POST http://localhost:3000/api/cron/process-goal-notifications \
  -H "Authorization: Bearer dev-secret-key"
```

Analytics & Metrics

Track goal notification performance:

Key Metrics:

1. **Confirmation Email Rate:** % of goals with confirmation sent
2. **Daily Check-in Open Rate:** % of daily emails opened
3. **Milestone Email CTR:** Click-through rate on achievement emails
4. **Reminder Completion:** % of users completing goals with daily reminders vs without
5. **Preference Adoption:** % of users enabling daily reminders
6. **Email Engagement:** Open rates, click rates, unsubscribe rates

Query Examples:

```

-- Goals with daily reminders enabled
SELECT COUNT(*)
FROM ScheduledNotification
WHERE type = 'GOAL_DAILY_CHECKIN'
    AND status = 'PENDING';

-- Milestone achievement emails sent
SELECT COUNT(*)
FROM EmailNotification
WHERE type = 'milestone_achieved'
    AND status = 'SENT';

-- Goal completion rate comparison
SELECT
    (SELECT COUNT(*) FROM Goal WHERE status = 'COMPLETED' AND has_reminders = true) * 10
0.0 /
    (SELECT COUNT(*) FROM Goal WHERE has_reminders = true) as with_reminders,
    (SELECT COUNT(*) FROM Goal WHERE status = 'COMPLETED' AND has_reminders = false) * 1
00.0 /
    (SELECT COUNT(*) FROM Goal WHERE has_reminders = false) as without_reminders;

```

Best Practices

For Users:

1. **Enable daily reminders** - 42% higher goal achievement rate
2. **Set realistic notification frequency** - Daily for short-term goals, weekly for long-term
3. **Choose optimal times** - Morning for motivation, evening for reflection
4. **Act on tips** - Each tip includes specific, actionable advice
5. **Track milestones** - Break big goals into smaller, achievable milestones

For Developers:

1. **Monitor email delivery** - Track send failures and bounce rates
2. **Rotate tip content** - Ensure variety prevents monotony
3. **Respect user preferences** - Always honor opt-out and frequency choices
4. **Handle timezone correctly** - Use user's local timezone for scheduling
5. **Graceful degradation** - System works even if emails fail

Troubleshooting

Common Issues

Issue: Daily emails not sending

- **Check:** Cron job is running (/api/cron/process-goal-notifications)
- **Verify:** ScheduledNotification records exist with status PENDING
- **Debug:** Check logs for error messages in notification processing

Issue: Wrong tip showing up

- **Check:** Day number calculation in processGoalDailyCheckIns
- **Verify:** Tip generator rotation logic
- **Debug:** Log tip category and day number

Issue: Emails going to spam

- **Check:** Email headers and SPF/DKIM setup
- **Verify:** Sender reputation
- **Fix:** Use reputable email service (Resend, SendGrid)

Issue: Milestone email not triggered

- **Check:** Milestone status update API call

- **Verify:** `sendMilestoneAchievedEmail` called in PATCH handler

- **Debug:** Check API logs for errors

Issue: Duplicate daily emails

- **Check:** Multiple cron jobs running

- **Verify:** Notification marked as SENT after sending

- **Fix:** Add idempotency checks, deduplicate scheduled notifications

Future Enhancements

Planned improvements for goal notifications:

1. AI-Generated Tips

- Personalized tips based on user's skill level
- Learning from user's progress patterns
- Context-aware suggestions

2. Progress Visualization

- Embedded charts in emails
- Visual progress bars
- Milestone timeline graphics

3. Smart Scheduling

- ML-based optimal send times
- Engagement pattern learning
- Adaptive frequency

4. Social Features

- Share goals with accountability partners
- Group goal challenges
- Peer motivation

5. Advanced Analytics

- Goal completion predictions
- Risk of abandonment alerts
- Performance insights

6. Video Integration

- Link video analysis to goals
- Progress clips in emails
- Before/after comparisons

7. Voice Notifications

- Audio motivational messages
- Coach Kai voice recordings
- Podcast-style tips

8. Gamification

- Streaks for daily check-ins
- Badges for milestone completion
- Leaderboards for goal achievement

Future Enhancements

- [] Push notification support (Firebase/OneSignal)
 - [] SMS notifications (Twilio)
 - [] In-app notification center UI with dropdown
 - [] A/B testing for email templates
 - [] Advanced analytics dashboard
 - [] Smart send time optimization (ML-based)
 - [] Multi-language support
 - [] Rich media in notifications
 - [] Notification templates builder for admins
 - [] Notification sound customization
 - [] Do Not Disturb mode
 - [] Notification preview in preferences UI
 - [] Bulk actions (disable all, enable all)
 - [] Import/export preferences
-

Support

For issues or questions:

- Check test results: `npx tsx scripts/tests/test-notification-system.ts`
 - Review logs in production
 - Contact: support@mindfulchampion.com
-

Built with ❤️ by the Mindful Champion Team

Coach Kai Notification Integration

Overview

Coach Kai can now create reminders directly from natural language conversations! Users can simply ask Coach Kai to remind them about anything, and the system automatically:

- Parses the reminder request
- Extracts date/time information
- Categorizes the reminder
- Creates a scheduled notification
- Confirms back to the user

Architecture

Components

1. Reminder Parser (`lib/notifications/reminder-parser.ts`)

- Natural language processing for reminder detection
- Date/time extraction
- Frequency parsing

- Category inference
- Confidence scoring

2. Coach Kai Reminder Tool (lib/notifications/coach-kai-reminder-tool.ts)

- LLM integration layer
- Reminder creation orchestration
- Confirmation message generation
- User-friendly error handling

3. UI Components (components/coach/reminder-message-card.tsx)

- In-chat reminder confirmation display
- Quick actions (edit, delete)
- Links to notification settings
- Beautiful visual feedback

Natural Language Patterns

The system recognizes a wide variety of reminder requests:

Basic Reminders

"Remind me to practice serves tomorrow at 3 PM"
 "Set a reminder for my tournament next week"
 "Can you remind me to upload a video in 2 hours?"

Daily/Recurring Reminders

"Send me a daily reminder at 8 AM to review my goals"
 "I want to be reminded every morning to check my progress"
 "Remind me every Monday at 9 AM about tournaments"

Time-of-Day Reminders

"Remind me tomorrow morning to do my stretches"
 "Set a reminder for this evening at 6 PM"
 "Daily motivation at 7 AM please"

Category-Specific

"Notify me when to analyze my video tomorrow"
 "Remind me about the tournament registration next Friday"
 "Alert me to check match schedule on Wednesday"

Implementation Details

1. Reminder Detection

The API endpoint checks for reminder intent in user messages:

```
// In app/api/ai-coach/route.ts
const isReminderRequest = hasReminderIntent(message);

if (isReminderRequest) {
  reminderResult = await createReminderFromCoachKai({
    userId: session.user.id,
    userMessage: message,
    conversationId: conversation.id,
  });
}
```

2. Natural Language Parsing

The reminder parser extracts structured data from natural language:

```
// Example input: "Remind me to practice serves tomorrow at 3 PM"
const parsed = parseReminderRequest(userMessage);

// Output:
{
  isReminder: true,
  title: "practice serves",
  category: "TRAINING_REMINDER",
  scheduledFor: Date("2025-12-04T15:00:00"),
  frequency: "CUSTOM",
  timeOfDay: "15:00",
  confidence: 0.9
}
```

3. Database Integration

Reminders are stored with source tracking:

```
await prisma.scheduledNotification.create({
  data: {
    userId,
    category: parsed.category,
    type: 'REMINDER',
    title: parsed.title,
    message: parsed.description || parsed.title,
    scheduledFor: parsed.scheduledFor,
    status: 'PENDING',
    deliveryMethod: 'APP',
    source: 'COACH_KAI', // ← Tracked as Coach Kai reminder
    data: {
      frequency: parsed.frequency,
      timeOfDay: parsed.timeOfDay,
      createdFrom: 'coach_kai',
      conversationId,
      originalMessage: userMessage,
    },
  },
});
```

4. User Confirmation

Coach Kai responds with a natural confirmation:

" Got it! I'll remind you to "practice serves" tomorrow at 3 PM."

You can manage this reminder **in** your [Notifications Settings](/settings/notifications)."

System Prompt Updates

The Coach Kai system prompt now includes:

REMINDER CAPABILITIES:

You can help users set reminders! When they ask to be reminded about something, acknowledge it naturally:

- "Got it! I'll remind you to practice serves tomorrow at 3 PM
- "Perfect! Set up a daily reminder for 8 AM to review your goals
- "Done! You'll get a reminder every Monday about tournaments

UI Display

In-Chat Confirmation Card

Reminders created from Coach Kai are displayed with:

- Creation status badge
- Reminder icon
- Scheduled date/time
- Category badge
- Frequency indicator
- Quick links to settings
- Edit/delete actions

Reminders Dashboard

In the Notifications Settings, Coach Kai reminders are marked with:

- **Purple badge:** "Coach Kai" source indicator
- **Bot icon:** Visual distinction from user-created reminders
- Full reminder details and management options

Testing

Pattern Detection Tests

Run the test suite to verify reminder detection:

```
node test-reminder-parser.js
```

Expected results:

- All reminder patterns detected correctly
- Non-reminder messages correctly ignored
- 100% accuracy on test cases

Test Cases Covered

1. Basic reminders with specific times
2. Daily/weekly recurring reminders
3. Time-of-day phrases (morning, evening, etc.)
4. Relative time expressions (tomorrow, next week, in 2 hours)

5. Category-specific language
6. Edge cases (non-reminder messages)

Usage Examples

Example 1: Quick Practice Reminder

User: "Remind me to practice my serves tomorrow at 3 PM"

Coach Kai: "Hey champ! 🎾 I'll totally remind you to practice serves tomorrow at 3 PM!"

That's a great focus - serves are where you build confidence. 💪

Got it! I'll remind you to "practice serves" tomorrow at 3 PM.

 You can manage this reminder in your Notifications Settings."

Example 2: Daily Goal Review

User: "Set a daily reminder at 8 AM to review my goals"

Coach Kai: "Love the commitment! 🔥 Daily goal reviews are game-changers!"

⏱️ Perfect! Set up a daily reminder for 8 AM to review your goals.

Starting tomorrow, you'll get a notification every morning. Stay consistent! 💪"

Example 3: Tournament Prep

User: "Remind me every Monday to check the tournament schedule"

Coach Kai: "Smart move! 🏆 Staying on top of tournaments helps you plan better!"

🏆 Done! You'll get a reminder every Monday about checking tournaments.

I'll ping you every Monday morning so you never miss a registration! 📡"

Best Practices

For Users:

1. **Be specific with times:** "tomorrow at 3 PM" works better than "sometime tomorrow"
2. **Mention frequency clearly:** Use "daily", "weekly", or specific days
3. **Add context:** More detail helps categorization ("practice serves" vs. just "practice")

For Developers:

1. **Monitor confidence scores:** Log low-confidence parses for improvement
2. **Track failed parses:** Use for pattern enhancement
3. **Test edge cases:** Regularly update test suite
4. **User feedback:** Add feedback mechanisms for incorrect parses

Analytics & Monitoring

Track Coach Kai reminder metrics:

```
// Get reminder statistics
const stats = await getUserReminderStats(userId);
// Returns: { active: 5, completed: 12, recent: [...], total: 17 }
```

Key Metrics:

- Total reminders created via Coach Kai
- Active vs completed reminders
- Most common reminder categories
- Parse confidence distribution
- User engagement rates

Future Enhancements

Planned improvements:

1. **Multi-language support:** Expand beyond English
2. **Smart suggestions:** Proactive reminder recommendations
3. **Context awareness:** Learn user patterns and preferences
4. **Voice integration:** Voice-to-reminder conversion
5. **Collaborative reminders:** Share reminders with practice partners
6. **Smart rescheduling:** AI-powered optimal timing suggestions

Troubleshooting

Common Issues

Issue: Reminder not detected

- **Solution:** Check patterns in `reminder-parser.ts`, add missing patterns
- **Debug:** Log `hasReminderIntent()` output

Issue: Wrong time parsed

- **Solution:** Verify timezone settings in user preferences
- **Debug:** Check `extractDateTime()` function output

Issue: Low confidence score

- **Solution:** Ask user to clarify with more specific language
- **Debug:** Review `calculateConfidence()` logic

Issue: Wrong category assigned

- **Solution:** Update category keywords in `CATEGORY_KEYWORDS`
- **Debug:** Log parsed category and user message

API Reference

`hasReminderIntent(message: string): boolean`

Quick check if message contains reminder intent.

`parseReminderRequest(message: string): ParsedReminder | null`

Full parsing of reminder request into structured data.

`createReminderFromCoachKai(input: ReminderToolInput): Promise<ReminderToolOutput>`

Main function to create reminder from chat.

```
getUserReminderStats(userId: string)
```

Get reminder statistics for a user.

```
cancelReminder(userId: string, reminderId: string): Promise<boolean>
```

Cancel a specific reminder.

```
updateReminder(userId: string, reminderId: string, updates: Partial<Reminder>): Promise<boolean>
```

Update reminder details.



Reminder Categories

Category Mapping

Category	Keywords	Icon	Use Case
GOALS	goal, objective, target, achievement	🎯	Goal-related reminders
VIDEO_ANALYSIS	video, analysis, analyze, upload, record	🎥	Video upload/review reminders
TOURNAMENTS	tournament, competition, match, game	🏆	Tournament prep and registration
TRAINING_Reminder	practice, drill, training, exercise	🏋️	Practice session reminders
MATCH_Reminder	match, play, court, opponent	🎾	Match and game reminders
MENTAL_TRAINING	mental, mindset, focus, meditation	🧠	Mental game reminders
MEDIA	watch, stream, podcast, content	📺	Media content reminders
COACH_KAI	ask coach, chat, question, advice	💬	Coach Kai follow-up reminders
GENERAL	(fallback)	🔔	General reminders

Source Tracking

All reminders created through Coach Kai are tagged with:

- **source: COACH_KAI**
- **data.createdFrom: coach_kai**
- **data.conversationId: Original conversation ID**
- **data.originalMessage: User's original message**

This enables:

- Filtering Coach Kai reminders
 - Conversation context tracking
 - Analytics and insights
 - User preference learning
-



Environment Variables Configuration

Critical Variables for Notifications

The notification system requires several environment variables to function properly. For complete setup instructions, see:

- [ENV_VARIABLES_CHECKLIST.md](#) ([./ENV_VARIABLES_CHECKLIST.md](#)) - Complete checklist with priority levels
- [GMAIL_SETUP.md](#) ([./GMAIL_SETUP.md](#)) - Detailed Gmail configuration guide
- [CRON_SECRET_DOCUMENTATION.md](#) ([./CRON_SECRET_DOCUMENTATION.md](#)) - Cron security documentation
- [SECURITY_BEST_PRACTICES.md](#) ([./SECURITY_BEST_PRACTICES.md](#)) - Security guidelines
- [DEPLOYMENT_GUIDE.md](#) ([./DEPLOYMENT_GUIDE.md](#)) - Platform-specific deployment instructions

Quick Reference

```
# =====
# NOTIFICATION SYSTEM - CRITICAL VARIABLES
# =====

# Gmail SMTP Configuration (CRITICAL)
GMAIL_USER=welcomefrommc@mindfulchampion.com
GMAIL_APP_PASSWORD=your-16-char-app-password

# Email Addresses for Different Purposes
NOTIFICATION_EMAIL=welcomefrommc@mindfulchampion.com
SUPPORT_EMAIL=support@mindfulchampion.com
PARTNERS_EMAIL=partners@mindfulchampion.com
SPONSORS_EMAIL=sponsors@mindfulchampion.com
EMAIL_FROM=welcomefrommc@mindfulchampion.com
EMAIL_REPLY_TO=dean@mindfulchampion.com

# Email Configuration
EMAIL_NOTIFICATIONS_ENABLED=true

# Cron Job Security (CRITICAL)
CRON_SECRET=jYEyBMds0rDgrmmAPi8yuUisg2C0zDfqnf/8/1VkJCo=

# Public App URL (CRITICAL)
NEXT_PUBLIC_APP_URL=https://mindfulchampion.com
```

Setup Priority

● CRITICAL (Must have for notifications to work):

1. GMAIL_USER - Gmail account for sending emails
2. GMAIL_APP_PASSWORD - Gmail SMTP authentication

3. CRON_SECRET - Protects cron endpoints
4. NEXT_PUBLIC_APP_URL - For links in emails

 **IMPORTANT (For complete functionality):**

1. NOTIFICATION_EMAIL - "From" address for notifications
2. SUPPORT_EMAIL - Support inquiry destination
3. EMAIL_FROM - Default sender email
4. EMAIL_REPLY_TO - Reply-to address
5. EMAIL_NOTIFICATIONS_ENABLED - Global toggle

 **OPTIONAL (Nice to have):**

1. PARTNERS_EMAIL - Partner communications
2. SPONSORS_EMAIL - Sponsor communications

Verification

Run the verification script to check your configuration:

```
npm run verify-env
```

This will:

- Check all required variables are set
- Validate Gmail credentials
- Test SMTP connection
- Verify CRON_SECRET format
- Report missing/invalid variables

Security Notes

 **NEVER commit these to version control:**

- GMAIL_APP_PASSWORD - Gives full access to Gmail account
- CRON_SECRET - Protects cron endpoints from abuse

Safe to expose:

- NEXT_PUBLIC_APP_URL - Public URL
- EMAIL_FROM - Public sender address
- NOTIFICATION_EMAIL - Public notification address

See [SECURITY_BEST_PRACTICES.md](#) ([./SECURITY_BEST_PRACTICES.md](#)) for complete security guidelines.

Gmail Email Configuration

Overview

The Mindful Champion notification system uses **Gmail SMTP** for reliable email delivery with support for **multiple sender addresses** for different types of communications.

Email Service Architecture

The Gmail email service is implemented in `/lib/email/gmail-service.ts` using **nodemailer** with connection pooling for optimal performance.

Sender Configurations

The system supports four distinct sender addresses:

Sender Type	Email	Name	Use Cases
NOTIFICATION	welcomefrom-mc@mindfulchampion.com	Coach Kai - Mindful Champion 🏆	Goal notifications, video analysis complete, training updates
SUPPORT	support@mindfulchampion.com	Mindful Champion Support 💬	Support inquiries, help desk responses
PARTNERSHIP	partners@mindfulchampion.com	Mindful Champion Partnerships 🤝	Partner communications, collaboration proposals
SPONSORSHIP	sponsors@mindfulchampion.com	Mindful Champion Sponsors 🎯	Sponsor applications and updates

Email Type Mapping

The email service automatically maps notification types to appropriate senders:

```
function mapEmailType(type: string): EmailType {
  switch (type) {
    case 'VIDEO_ANALYSIS_COMPLETE':
    case 'GOAL_RemINDER':
    case 'TRAINING_UPDATE':
      return 'NOTIFICATION';
    case 'SUPPORT':
    case 'HELP':
      return 'SUPPORT';
    case 'PARTNER_APPLICATION':
    case 'PARTNER_UPDATE':
      return 'PARTNERSHIP';
    case 'SPONSOR_APPLICATION':
    case 'SPONSOR_UPDATE':
      return 'SPONSORSHIP';
    default:
      return 'NOTIFICATION';
  }
}
```

Configuration

Environment Variables

Required Gmail SMTP configuration:

```
# Gmail SMTP Configuration
GMAIL_USER=your-gmail@gmail.com
GMAIL_APP_PASSWORD=your-16-char-app-password

# Email Addresses for Different Purposes
NOTIFICATION_EMAIL=welcomefrommc@mindfulchampion.com
SUPPORT_EMAIL=support@mindfulchampion.com
PARTNERS_EMAIL=partners@mindfulchampion.com
SPONSORS_EMAIL=sponsors@mindfulchampion.com

# Email Settings
EMAIL_NOTIFICATIONS_ENABLED=true
EMAIL_REPLY_TO=dean@mindfulchampion.com
```

SMTP Settings

- **Host:** smtp.gmail.com
- **Port:** 587 (TLS)
- **Secure:** false (uses STARTTLS)
- **Connection Pool:** Enabled (5 max connections, 100 max messages)
- **Authentication:** Gmail App Password (requires 2FA)

Usage

Basic Email Sending

```
import { sendEmail } from '@/lib/email/gmail-service';

// Send a notification email
const result = await sendEmail({
  to: 'user@example.com',
  subject: 'Your video analysis is ready!',
  html: '<p>Your analysis is complete.</p>',
  type: 'NOTIFICATION',
});

if (result.success) {
  console.log('Email sent:', result.messageId);
}
```

Using Convenience Functions

```
import {
  sendNotificationEmail,
  sendSupportEmail,
  sendPartnershipEmail,
  sendSponsorshipEmail
} from '@/lib/email/gmail-service';

// Send notification email (uses NOTIFICATION sender)
await sendNotificationEmail(
  'user@example.com',
  'Goal Reminder',
  '<p>Time to work on your backhand!</p>'
);

// Send support email (uses SUPPORT sender)
await sendSupportEmail(
  'user@example.com',
  'Support Inquiry Response',
  '<p>Thanks for reaching out...</p>'
);
```

With Email Service (Database Integration)

```
import { emailService } from '@/lib/email/email-service';

// Send email with database tracking
const result = await emailService.sendEmail({
  userId: 'user_123',
  recipientEmail: 'user@example.com',
  subject: 'Video Analysis Complete',
  htmlContent: '<p>Your analysis is ready!</p>',
  type: 'VIDEO_ANALYSIS_COMPLETE', // Automatically maps to NOTIFICATION sender
  videoAnalysisId: 'analysis_456',
});
```

Gmail Setup Guide

For detailed Gmail setup instructions, see: [GMAIL_SETUP.md](#) (/GMAIL_SETUP.md)

Quick setup steps:

1. **Enable 2-Factor Authentication** on your Gmail account
2. **Generate App Password** at [Google App Passwords](https://myaccount.google.com/apppasswords) (<https://myaccount.google.com/apppasswords>)
3. **Configure .env** with `GMAIL_USER` and `GMAIL_APP_PASSWORD`
4. **Test configuration** with: `npx tsx scripts/test-gmail.ts`

Testing

Test Script

Run the comprehensive Gmail test script:

```
cd /home/ubuntu/mindful_champion/nextjs_space
npx tsx scripts/test-gmail.ts your-test-email@example.com
```

The script will:

- Verify Gmail connection
- Check environment variables
- Test all sender configurations
- Send test emails from each sender

Verify Connection

```
import { verifyGmailConnection } from '@/lib/email/gmail-service';

const isConnected = await verifyGmailConnection();
console.log('Gmail connected:', isConnected);
```

Features

Connection Pooling

The Gmail service uses connection pooling for better performance:

- **5 concurrent connections** maximum
- **100 messages per connection** before refresh
- **Automatic reconnection** on failures
- **Connection verification** on startup

Error Handling

All email operations include:

- **Database tracking** via `EmailNotification` records
- **Retry mechanism** with configurable attempts
- **Failed status tracking** with error messages
- **Video analysis status updates** for analysis emails

Email Status Tracking

Emails are tracked with the following statuses:

- `PENDING` - Email queued for sending
- `SENDING` - Email currently being sent
- `SENT` - Email successfully delivered
- `FAILED` - Email delivery failed
- `OPENED` - User opened the email (if tracking enabled)

Sending Limits

Gmail imposes daily sending limits:

Account Type	Limit
Free Gmail	500 emails/day
Google Workspace	2,000 emails/day

Recommendation: Use Google Workspace for production applications.

Deliverability Best Practices

To ensure emails reach inboxes:

1. **SPF Record:** Add to DNS for your domain

```
v=spf1 include:_spf.google.com ~all
```

2. **DKIM** (Google Workspace only): Enable in Google Admin Console

3. **Email Content:**

- Include both HTML and plain text versions
- Avoid spam trigger words
- Include unsubscribe link
- Use consistent “From” name

4. **Sender Reputation:**

- Monitor bounce rates
- Handle unsubscribes promptly
- Remove invalid addresses
- Track engagement metrics

Monitoring & Maintenance

Email Statistics

Track email performance via `emailService.getEmailStats()`:

```
const stats = await emailService.getEmailStats();
console.log({
  total: stats.total,
  sent: stats.sent,
  failed: stats.failed,
  successRate: `${stats.successRate}%`
});
```

Failed Email Retry

Automatically retry failed emails:

```
const result = await emailService.retryEmail(emailNotificationId);
```

Cleanup

Close the transporter on application shutdown:

```
import { closeGmailTransporter } from '@/lib/email/gmail-service';

// On app shutdown
closeGmailTransporter();
```

Troubleshooting

Common Issues

Issue	Solution
“Invalid login”	Verify 2FA is enabled and App Password is correct
“Connection timeout”	Check firewall allows port 587 outbound
“Daily limit exceeded”	Wait 24 hours or upgrade to Google Workspace
Emails in spam	Set up SPF/DKIM, improve content
“Mock transporter” logs	Check <code>.env</code> has <code>GMAIL_USER</code> and <code>GMAIL_APP_PASSWORD</code>

For detailed troubleshooting, see [GMAIL_SETUP.md](#) (/GMAIL_SETUP.md).

Migration from Resend

The system has been updated to use Gmail instead of Resend:

- All `resend.emails.send()` calls replaced with `sendEmail()`
- Email notification tracking preserved
- Video analysis email status updates maintained
- Retry mechanism updated
- Statistics tracking functional

Legacy Resend configuration is preserved but not used when Gmail is configured.

UI/UX Considerations

Design Principles

1. **Natural Integration:** Reminders feel like a natural part of the conversation
2. **Clear Feedback:** Users always know when a reminder is created
3. **Easy Management:** Quick access to edit/delete reminders
4. **Visual Distinction:** Coach Kai reminders are clearly marked
5. **Consistent Branding:** Uses Coach Kai’s personality and tone

Accessibility

- Screen reader friendly
- Keyboard navigation support
- High contrast mode support
- Clear visual indicators
- Descriptive ARIA labels

Mobile Optimization

- Touch-friendly buttons
 - Responsive card layouts
 - Swipe actions for edit/delete
 - Native notification support
-

Security & Privacy

Data Protection

- User reminders are private and not shared
- Encryption for sensitive data
- Secure API endpoints (session-based auth)
- GDPR compliance

Permissions

- Users control all reminder settings
 - Can disable Coach Kai reminders
 - Full deletion capabilities
 - Export data on request
-

Success Metrics

Track these KPIs:

1. **Creation Rate:** Reminders created per conversation
 2. **Parse Accuracy:** Successful vs failed parses
 3. **User Retention:** Users who create multiple reminders
 4. **Engagement:** Reminder open/interaction rates
 5. **Satisfaction:** User feedback on reminder accuracy
-

Testing

Test Suite Overview

A comprehensive test suite has been created for the notification system. See [TESTING_REPORT.md](#) (./TESTING_REPORT.md) for detailed information.

Test Coverage: 42 tests across 7 modules

-  Notification Preferences (5 tests)
-  Reminders Dashboard (6 tests)
-  Coach Kai Integration (7 tests)
-  Goal Notifications (7 tests)
-  Email Delivery (5 tests)

- Cron Jobs (7 tests)
- Integration Tests (5 tests)

Running Tests

```
# Run all tests
cd scripts/tests/notification-system
npx ts-node run-all-tests.ts

# Run specific test module
npx ts-node preferences-tests.ts
npx ts-node reminders-tests.ts
npx ts-node coach-kai-tests.ts
# ... etc
```

Test Requirements

1. **Database:** Running and accessible
2. **Email:** Gmail SMTP credentials configured
3. **Application:** Development or production server running
4. **Test User:** User with email containing 'test'

Continuous Testing

- Tests should be run before each deployment
- Automated testing recommended for CI/CD
- Monitor test coverage and execution time
- Update tests when adding new features



Deployment Checklist

Before going live:

- [x] Reminder parser tested with 100+ patterns
- [x] API endpoints secured with authentication
- [x] Database migrations applied
- [x] UI components responsive and accessible
- [x] Error handling implemented
- [x] Analytics tracking configured
- [x] Documentation completed
- [x] Comprehensive test suite created (42 tests)
- [] Execute full test suite and verify all pass
- [] User acceptance testing
- [] Load testing for high volume
- [] Monitoring and alerts set up

Last Updated: December 3, 2025

Version: 2.1.0

Contributors: Development Team