* + // Notification System Implementation
  + interface NotificationPreferences {
  + emailEnabled: boolean;
  + inAppEnabled: boolean;
  + reminderSchedule: 'default' | 'custom';
  + customSchedule?: number[]; // Days after posting
  + excludedTypes?: string[]; // Types of notifications to exclude
  + }
  + interface NotificationTrigger {
  + type: 'post\_created' | 'time\_based' | 'milestone' | 'system';
  + postId?: string;
  + userId: string;
  + schedule?: number[];
  + data?: Record<string, any>;
  + }
  + class NotificationManager {
  + async scheduleNotifications(trigger: NotificationTrigger): Promise<void> {
  + const userPrefs = await this.getUserPreferences(trigger.userId);
  + if (trigger.type === 'post\_created') {
  + // Schedule default reminder sequence
  + await this.schedulePostReminders(trigger.postId, userPrefs);
  + }
  + }
  + private async schedulePostReminders(postId: string, prefs: NotificationPreferences): Promise<void> {
  + const reminderSchedule = prefs.reminderSchedule === 'custom'
  + ? prefs.customSchedule
  + : [1, 7, 30]; // Default schedule (days)
  + for (const day of reminderSchedule) {
  + await this.createReminderJob({
  + postId,
  + scheduledFor: addDays(new Date(), day),
  + channels: this.getEnabledChannels(prefs)
  + });
  + }
  + }
  + private async sendNotification(notification: Notification): Promise<void> {
  + if (notification.channels.includes('email')) {
  + await this.sendEmailNotification(notification);
  + }
  + if (notification.channels.includes('inApp')) {
  + await this.sendInAppNotification(notification);
  + }
  + }
  + private async sendEmailNotification(notification: Notification): Promise<void> {
  + const emailTemplate = this.getEmailTemplate(notification.type);
  + await resend.emails.send({
  + from: 'Ghostart <updates@ghostart.com>',
  + to: notification.userEmail,
  + subject: emailTemplate.subject,
  + react: emailTemplate.component(notification.data)
  + });
  + }
  + }
  + // Email Templates
  + const PerformanceUpdateEmail = ({ postData }) => (
  + <EmailTemplate>
  + <Heading>Time to Update Your Post Performance</Heading>
  + <Text>
  + It's been {postData.daysAgo} days since you published your post.
  + Take a moment to update its performance metrics.
  + </Text>
  + <Button href={postData.updateUrl}>
  + Update Performance
  + </Button>
  + <PostPreview post={postData} />
  + </EmailTemplate>
  + );
  + // Notification Queue Handler
  + class NotificationQueue {
  + async processNotificationJobs(): Promise<void> {
  + const jobs = await this.getScheduledNotifications();
  + for (const job of jobs) {
  + try {
  + await this.processNotification(job);
  + await this.markAsProcessed(job.id);
  + } catch (error) {
  + await this.handleNotificationError(job, error);
  + }
  + }
  + }
  + }
  + // Database Schema
  + const notificationSchema = `
  + -- Notification Preferences
  + CREATE TABLE notification\_preferences (
  + user\_id UUID REFERENCES auth.users PRIMARY KEY,
  + email\_enabled BOOLEAN DEFAULT true,
  + in\_app\_enabled BOOLEAN DEFAULT true,
  + reminder\_schedule TEXT DEFAULT 'default',
  + custom\_schedule INTEGER[],
  + excluded\_types TEXT[],
  + updated\_at TIMESTAMPTZ DEFAULT NOW()
  + );
  + -- Notification Queue
  + CREATE TABLE notification\_queue (
  + id UUID DEFAULT uuid\_generate\_v4() PRIMARY KEY,
  + user\_id UUID REFERENCES auth.users,
  + type TEXT NOT NULL,
  + scheduled\_for TIMESTAMPTZ NOT NULL,
  + data JSONB,
  + status TEXT DEFAULT 'pending',
  + created\_at TIMESTAMPTZ DEFAULT NOW()
  + );
  + -- Notification History
  + CREATE TABLE notification\_history (
  + id UUID DEFAULT uuid\_generate\_v4() PRIMARY KEY,
  + notification\_id UUID REFERENCES notification\_queue,
  + status TEXT NOT NULL,
  + sent\_at TIMESTAMPTZ DEFAULT NOW(),
  + channel TEXT NOT NULL,
  + error TEXT
  + );
  + `;

Key Features:

1. Multiple notification channels (email, in-app)
2. Customizable reminder schedule
3. Smart triggers based on post activity
4. User preferences management
5. Error handling and retry logic