Đánh giá và cải thiện kế hoạch triển khai tính năng thông báo Telegram

Bạn có nhận xét đúng - các prompt và kế hoạch phát triển ở trên còn khá chung chung và thiếu chi tiết cụ thể để Chat Bot có thể hiểu và thực hiện từng bước một cách hiệu quả. Dưới đây là phiên bản chi tiết và cụ thể hơn:

I. Chuẩn bị và khảo sát hệ thống

Prompt 1: Phân tích cấu trúc database hiện tại

```
Hãy phân tích file schema.txt và trigger_functions.txt để:
1. Liệt kê chi tiết cấu trúc bảng profiles hiện tại và các field liên quan
đến notification
2. Xác định cấu trúc bảng notifications hiện tại (nếu có)
3. Phân tích các triggers và functions hiện có liên quan đến thông báo
4. Chỉ ra chính xác các trường cần thêm cho tính năng Telegram
5. Dựa vào cấu trúc database hiện tại, đề xuất các thay đổi cần thực hiện
Đánh giá xem đã có telegram_id trong bảng profiles chưa và cách thức tích
hợp tối ưu nhất.
```

Prompt 2: Phân tích luồng dữ liệu thông báo hiện tại

```
Dựa trên file code.txt, hãy:
1. Phân tích kỹ cách hệ thống hiện tại xử lý notifications
2. Xác định chính xác cấu trúc thư mục, các file và components liên quan
đến thông báo
3. Tìm các API routes liên quan đến notifications (/api/notifications/...)
4. Phân tích các React hooks, queries và mutations đang được sử dụng cho
notifications
5. Xác định chính xác các utils hiện có để xử lý notifications
6. Liệt kê chi tiết các cấu trúc dữ liệu, types, và interfaces liên quan

Sử dụng thông tin này để xác định cách tích hợp thông báo Telegram vào hệ
thống hiện tại một cách mượt mà nhất.
```

II. Backend (Supabase)

Prompt 3: Cập nhật Schema Database

```
Dựa trên phân tích schema.txt và cấu trúc database hiện tại, hãy viết SQL
code chính xác để:

1. Thêm các trường sau vào bảng profiles (CHÍNH XÁC, đảm bảo kiểu dữ liệu
```

```
đúng):
   - telegram_id TEXT
   - telegram_username TEXT
   - telegram_connected_at TIMESTAMPTZ
   - telegram_settings JSONB DEFAULT '{"account": true, "payment": true,
"bet": true, "system": true}'

2. Tạo bảng mới telegram_notifications với cấu trúc:
   CREATE TABLE public.telegram_notifications (
       id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
       user_id UUID NOT NULL REFERENCES public.profiles(id) ON DELETE
CASCADE,
       message TEXT NOT NULL,
       metadata JSONB DEFAULT '{}',
       sent_at TIMESTAMPTZ DEFAULT now(),
       status TEXT DEFAULT 'pending',
       created_at TIMESTAMPTZ DEFAULT now()
   );

3. Tạo indexes cần thiết:
   CREATE INDEX idx_telegram_notifications_user_id ON
public.telegram_notifications(user_id);
   CREATE INDEX idx_telegram_notifications_status ON
public.telegram_notifications(status);
   CREATE INDEX idx_telegram_notifications_sent_at ON
public.telegram_notifications(sent_at);

Đảm bảo code SQL hoàn toàn chính xác về cú pháp và phù hợp với PostgreSQL
trong Supabase.
```

Prompt 4: Tạo Function gửi thông báo Telegram

```
Viết một function SQL đầy đủ và chi tiết trong Supabase để gửi thông báo
qua Telegram Bot API. Function này phải:

1. Có signature: create_telegram_notification(p_user_id UUID, p_message
TEXT, p_metadata JSONB DEFAULT '{}')
2. Kiểm tra xem user có telegram_id không, nếu không thì RETURN ngay
3. Kiểm tra cài đặt telegram_settings của user xem có cho phép gửi loại
thông báo này không (dựa vào p_metadata->'type')
4. Tạo bản ghi trong bảng telegram_notifications
5. Sử dụng pg_net extension để gửi HTTP request đến Telegram Bot API:
   - URL: https://api.telegram.org/bot{{BOT_TOKEN}}/sendMessage
   - Method: POST
   - Headers: Content-Type: application/json
   - Body: JSON với chat_id và text

Function phải bao gồm đầy đủ:
- Error handling
- Logging
- Return type rõ ràng (SETOF telegram_notifications)
```

```
- Comments giải thích code

Đây là template cơ bản:

CREATE OR REPLACE FUNCTION public.create_telegram_notification(...)
RETURNS SETOF public.telegram_notifications
LANGUAGE plpgsql
SECURITY DEFINER
AS $$
DECLARE
  v_telegram_id TEXT;
  v_telegram_settings JSONB;
  v_notification_type TEXT;
  v_should_send BOOLEAN := true;
  v_response_id UUID;
  v_notification_id UUID;
BEGIN
  -- Get user's telegram_id
  ...

  -- Check notification settings
  ...

  -- Create notification record
  ...

  -- Send to Telegram API using pg_net
  ...

  RETURN QUERY SELECT * FROM public.telegram_notifications WHERE id =
v_notification_id;
END;
$$;
```

Prompt 5: Tạo các Trigger functions chi tiết

```
Dựa trên phân tích trigger_functions.txt, hãy viết các trigger functions
cụ thể để gửi thông báo Telegram cho các sự kiện quan trọng:

1. CREATE OR REPLACE FUNCTION public.on_payment_request_status_change()
cho bảng payment_requests
   - Kích hoạt khi status thay đổi (NEW.status != OLD.status)
   - Gửi thông báo khi status = 'approved' hoặc 'rejected'
   - Format nội dung thông báo khác nhau cho nạp tiền vs rút tiền

2. CREATE OR REPLACE FUNCTION public.on_bet_resolved() cho bảng bets
   - Kích hoạt khi status thay đổi thành 'won' hoặc 'lost'
   - Gửi thông báo với chi tiết kết quả, số tiền cược và số tiền thắng
   - Chỉ gửi cho cược thắng với số tiền lớn (> 1,000,000)

3. CREATE OR REPLACE FUNCTION public.on_account_status_change()
```

```
   - Theo dõi thay đổi trạng thái tài khoản (khóa/mở khóa)
   - Gửi thông báo ngay lập tức khi có thay đổi

Mỗi function cần có đầy đủ:
- Parameter đúng cú pháp PostgreSQL trigger
- Logic kiểm tra điều kiện cụ thể trước khi gửi
- Gọi function create_telegram_notification với tham số chính xác
- Cú pháp CREATE TRIGGER để gắn trigger vào các bảng

Ví dụ template cho function on_payment_request_status_change():

CREATE OR REPLACE FUNCTION public.on_payment_request_status_change()
RETURNS TRIGGER
LANGUAGE plpgsql
SECURITY DEFINER
AS $$
DECLARE
  v_message TEXT;
  v_metadata JSONB;
BEGIN
  -- Chỉ xử lý khi status thay đổi
  IF NEW.status = OLD.status THEN
    RETURN NEW;
  END IF;

  -- Xử lý khi payment request được approved
  IF NEW.status = 'approved' THEN
    -- Format message based on type (deposit/withdrawal)
    IF NEW.type = 'deposit' THEN
      v_message := 'Nạp tiền thành công: ' || format_money(NEW.amount) ||
'. Số dư hiện tại: ' || ...;
    ELSE
      v_message := 'Rút tiền thành công: ' || format_money(NEW.amount) ||
'. Số dư hiện tại: ' || ...;
    END IF;

    v_metadata := jsonb_build_object(
      'type', 'payment',
      'payment_id', NEW.id,
      'amount', NEW.amount,
      'payment_type', NEW.type
    );

    -- Gửi thông báo
    PERFORM create_telegram_notification(NEW.user_id, v_message,
v_metadata);
  END IF;

  -- Xử lý khi payment request bị từ chối
  IF NEW.status = 'rejected' THEN
    ...
  END IF;

  RETURN NEW;
```

```
END;
$$;

CREATE TRIGGER trigger_payment_request_status_change
AFTER UPDATE ON public.payment_requests
FOR EACH ROW
EXECUTE FUNCTION public.on_payment_request_status_change();
```

Prompt 6: Thiết lập RLS Policies chi tiết

```
Dựa trên phân tích policies.txt, hãy viết SQL chính xác để thiết lập Row
Level Security (RLS) cho bảng telegram_notifications:

1. Bật RLS trên bảng telegram_notifications:
   ALTER TABLE public.telegram_notifications ENABLE ROW LEVEL SECURITY;

2. Tạo policy cho phép users đọc thông báo của chính mình:
   CREATE POLICY "Users can view their own telegram notifications"
   ON public.telegram_notifications
   FOR SELECT
   TO authenticated
   USING (auth.uid() = user_id);

3. Tạo policy cho phép admins đọc tất cả thông báo:
   CREATE POLICY "Admins can view all telegram notifications"
   ON public.telegram_notifications
   FOR SELECT
   TO authenticated
   USING (EXISTS (
     SELECT 1 FROM public.profiles
     WHERE profiles.id = auth.uid() AND profiles.role = 'admin'
   ));

4. Tạo policy cho phép admins cập nhật thông báo:
   CREATE POLICY "Admins can update telegram notifications"
   ON public.telegram_notifications
   FOR UPDATE
   TO authenticated
   USING (EXISTS (
     SELECT 1 FROM public.profiles
     WHERE profiles.id = auth.uid() AND profiles.role = 'admin'
   ));

5. Thiết lập chính sách cho các trường telegram_* trong bảng profiles:
   (Chỉ cho phép users cập nhật telegram settings của chính mình)

Đảm bảo code SQL hoàn toàn chính xác và tuân thủ cú pháp PostgreSQL
policy.
```

III. Frontend và NextJS Implementation

Prompt 7: Tạo Bot Telegram và Webhook setup

````
Tạo hướng dẫn chi tiết và code triển khai để tạo Telegram Bot và thiết lập
webhook:

1. Chi tiết từng bước tạo bot trên Telegram sử dụng BotFather:
   - Sử dụng lệnh /newbot trong BotFather
   - Đặt tên và username cho bot
   - Lưu lại API token

2. Code đầy đủ cho API route webhook của Telegram trong NextJS:

File: src/app/api/telegram/webhook/route.ts
```typescript
import { createRouteHandlerClient } from '@supabase/auth-helpers-nextjs';
import { cookies } from 'next/headers';
import { NextResponse } from 'next/server';

// Xác minh đây là request từ Telegram
function verifyTelegramWebhook(payload: any, botToken: string): boolean {
  // Implementation code here
}

export async function POST(request: Request) {
  try {
    const payload = await request.json();
    const botToken = process.env.TELEGRAM_BOT_TOKEN;

    if (!verifyTelegramWebhook(payload, botToken!)) {
      return NextResponse.json({ error: 'Unauthorized' }, { status: 401
});
    }

    // Xử lý các loại message từ Telegram
    if (payload.message?.text) {
      const { text, from } = payload.message;
      const chatId = from.id;
      const username = from.username;

      // Handle commands
      if (text === '/start') {
        // Respond with welcome message and verification instructions
      } else if (text.startsWith('/verify_')) {
        // Extract verification code and verify user
        const code = text.replace('/verify_', '');

        const supabase = createRouteHandlerClient({ cookies });

        // Find user with matching verification code
        const { data: userData, error } = await supabase
          .from('profiles')
          .select('id, telegram_verification_code')
````
````

```
          .eq('telegram_verification_code', code)
          .single();

        if (userData) {
          // Update user's telegram info
          await supabase
            .from('profiles')
            .update({
              telegram_id: chatId.toString(),
              telegram_username: username,
              telegram_connected_at: new Date().toISOString(),
              telegram_verification_code: null
            })
            .eq('id', userData.id);

          // Send confirmation
          await
fetch(`https://api.telegram.org/bot${botToken}/sendMessage`, {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify({
              chat_id: chatId,
              text: 'Kết nối thành công với tài khoản VinBet! Bạn sẽ nhận
được thông báo quan trọng qua Telegram.'
            })
          });
        }
      }
    }

    return NextResponse.json({ success: true });
  } catch (error) {
    console.error('Telegram webhook error:', error);
    return NextResponse.json({ error: 'Internal Server Error' }, { status:
500 });
  }
}
```

3. Hướng dẫn thiết lập webhook URL trên Telegram sử dụng setWebhook API:

   - Cú pháp: https://api.telegram.org/bot{BOT_TOKEN}/setWebhook?url={WEBHOOK_URL}
   - URL webhook phải có HTTPS
   - Cân nhắc sử dụng secret token để tăng cường bảo mật

4. Code test webhook để xác nhận hoạt động đúng

```
### Prompt 8: Tạo API Routes cho Telegram trong NextJS
```

Tạo các API Routes đầy đủ trong NextJS để quản lý kết nối Telegram:

1. API generate verification code:

File: src/app/api/telegram/generate-code/route.ts

```
import { createRouteHandlerClient } from '@supabase/auth-helpers-nextjs';
import { cookies } from 'next/headers';
import { NextResponse } from 'next/server';
import { randomUUID } from 'crypto';

export async function POST(request: Request) {
  try {
    const supabase = createRouteHandlerClient({ cookies });

    // Verify user is authenticated
    const { data: { user } } = await supabase.auth.getUser();
    if (!user) {
      return NextResponse.json({ error: 'Unauthorized' }, { status: 401
});
    }

    // Generate verification code (short and user-friendly)
    const verificationCode = randomUUID().substring(0, 8);

    // Store in database
    const { error } = await supabase
      .from('profiles')
      .update({ telegram_verification_code: verificationCode })
      .eq('id', user.id);

    if (error) {
      console.error('Error generating code:', error);
      return NextResponse.json({ error: 'Failed to generate code' }, {
status: 500 });
    }

    const botUsername = process.env.TELEGRAM_BOT_USERNAME;
    const deepLink = `https://t.me/${botUsername}?
start=verify_${verificationCode}`;

    return NextResponse.json({
      verificationCode,
      deepLink,
      botUsername
    });
  } catch (error) {
    console.error('Error in generate-code:', error);
    return NextResponse.json({ error: 'Internal Server Error' }, { status:
500 });
  }
}
```

2. API lấy trạng thái kết nối Telegram:

File: src/app/api/telegram/status/route.ts

```
import { createRouteHandlerClient } from '@supabase/auth-helpers-nextjs';
import { cookies } from 'next/headers';
import { NextResponse } from 'next/server';

export async function GET(request: Request) {
  try {
    const supabase = createRouteHandlerClient({ cookies });

    // Verify user is authenticated
    const { data: { user } } = await supabase.auth.getUser();
    if (!user) {
      return NextResponse.json({ error: 'Unauthorized' }, { status: 401
});
    }

    // Get telegram connection status
    const { data, error } = await supabase
      .from('profiles')
      .select('telegram_id, telegram_username, telegram_connected_at,
telegram_settings')
      .eq('id', user.id)
      .single();

    if (error) {
      console.error('Error fetching telegram status:', error);
      return NextResponse.json({ error: 'Failed to fetch status' }, {
status: 500 });
    }

    const isConnected = !!data.telegram_id;

    return NextResponse.json({
      isConnected,
      username: data.telegram_username,
      connectedAt: data.telegram_connected_at,
      settings: data.telegram_settings || {
        account: true,
        payment: true,
        bet: true,
        system: true
      }
    });
  } catch (error) {
    console.error('Error in status route:', error);
    return NextResponse.json({ error: 'Internal Server Error' }, { status:
500 });
  }
}
```

3. API cập nhật cài đặt thông báo Telegram:

File: src/app/api/telegram/settings/route.ts

```
import { createRouteHandlerClient } from '@supabase/auth-helpers-nextjs';
import { cookies } from 'next/headers';
import { NextResponse } from 'next/server';

export async function PUT(request: Request) {
  try {
    const supabase = createRouteHandlerClient({ cookies });

    // Verify user is authenticated
    const { data: { user } } = await supabase.auth.getUser();
    if (!user) {
      return NextResponse.json({ error: 'Unauthorized' }, { status: 401
});
    }

    // Get settings from request
    const { settings } = await request.json();

    // Validate settings
    const validSettings = {
      account: !!settings.account,
      payment: !!settings.payment,
      bet: !!settings.bet,
      system: !!settings.system
    };

    // Update in database
    const { data, error } = await supabase
      .from('profiles')
      .update({ telegram_settings: validSettings })
      .eq('id', user.id)
      .select('telegram_settings');

    if (error) {
      console.error('Error updating telegram settings:', error);
      return NextResponse.json({ error: 'Failed to update settings' }, {
status: 500 });
    }

    return NextResponse.json({
      settings: data[0].telegram_settings
    });
  } catch (error) {
    console.error('Error in settings route:', error);
    return NextResponse.json({ error: 'Internal Server Error' }, { status:
500 });
```

```
  }
}
```

4. API ngắt kết nối Telegram:

File: src/app/api/telegram/disconnect/route.ts

```
import { createRouteHandlerClient } from '@supabase/auth-helpers-nextjs';
import { cookies } from 'next/headers';
import { NextResponse } from 'next/server';

export async function POST(request: Request) {
  try {
    const supabase = createRouteHandlerClient({ cookies });

    // Verify user is authenticated
    const { data: { user } } = await supabase.auth.getUser();
    if (!user) {
      return NextResponse.json({ error: 'Unauthorized' }, { status: 401
});
    }

    // Clear telegram connection data
    const { error } = await supabase
      .from('profiles')
      .update({
        telegram_id: null,
        telegram_username: null,
        telegram_connected_at: null
      })
      .eq('id', user.id);

    if (error) {
      console.error('Error disconnecting telegram:', error);
      return NextResponse.json({ error: 'Failed to disconnect' }, {
status: 500 });
    }

    return NextResponse.json({
      success: true,
      message: 'Telegram disconnected successfully'
    });
  } catch (error) {
    console.error('Error in disconnect route:', error);
    return NextResponse.json({ error: 'Internal Server Error' }, { status:
500 });
  }
}
```

```
### Prompt 9: Tạo React Hooks cho Telegram
```

Tạo React Hooks đầy đủ và chi tiết cho quản lý thông báo Telegram, sử dụng React Query:

File: src/hooks/useTelegramConnection.ts

```
import { useMutation, useQuery, useQueryClient } from '@tanstack/react-
query';
import axios from 'axios';

// Types
export interface TelegramStatus {
  isConnected: boolean;
  username: string | null;
  connectedAt: string | null;
  settings: TelegramSettings;
}

export interface TelegramSettings {
  account: boolean;
  payment: boolean;
  bet: boolean;
  system: boolean;
}

// Hook chính
export function useTelegramConnection() {
  const queryClient = useQueryClient();

  // Fetch trạng thái kết nối
  const statusQuery = useQuery<TelegramStatus>({
    queryKey: ['telegram', 'status'],
    queryFn: async () => {
      const { data } = await axios.get('/api/telegram/status');
      return data;
    },
    refetchOnWindowFocus: true,
    staleTime: 1000 * 60 * 5, // 5 minutes
  });

  // Generate verification code
  const generateCodeMutation = useMutation({
    mutationFn: async () => {
      const { data } = await axios.post('/api/telegram/generate-code');
      return data;
    },
    onSuccess: () => {
      // Start polling for connection status
      startPolling();
```

```
    },
  });

  // Update settings
  const updateSettingsMutation = useMutation({
    mutationFn: async (settings: TelegramSettings) => {
      const { data } = await axios.put('/api/telegram/settings', {
settings });
      return data;
    },
    onSuccess: () => {
      queryClient.invalidateQueries({ queryKey: ['telegram', 'status'] });
    },
  });

  // Disconnect Telegram
  const disconnectMutation = useMutation({
    mutationFn: async () => {
      const { data } = await axios.post('/api/telegram/disconnect');
      return data;
    },
    onSuccess: () => {
      queryClient.invalidateQueries({ queryKey: ['telegram', 'status'] });
    },
  });

  // Polling function to check connection status
  const startPolling = () => {
    // Implementation details for polling
    const intervalId = setInterval(() => {
      queryClient.invalidateQueries({ queryKey: ['telegram', 'status'] });

      // Check if connected and stop polling if true
      const currentStatus = queryClient.getQueryData<TelegramStatus>
(['telegram', 'status']);
      if (currentStatus?.isConnected) {
        clearInterval(intervalId);
      }
    }, 3000); // Poll every 3 seconds

    // Clean up after 2 minutes (safety)
    setTimeout(() => clearInterval(intervalId), 2 * 60 * 1000);

    return intervalId;
  };

  return {
    // Trạng thái
    status: statusQuery.data,
    isConnected: statusQuery.data?.isConnected || false,
    isLoading: statusQuery.isLoading,
    error: statusQuery.error,

    // Mutations
```

```
    generateCode: generateCodeMutation.mutate,
    isGeneratingCode: generateCodeMutation.isPending,
    verificationData: generateCodeMutation.data,

    updateSettings: updateSettingsMutation.mutate,
    isUpdatingSettings: updateSettingsMutation.isPending,

    disconnect: disconnectMutation.mutate,
    isDisconnecting: disconnectMutation.isPending,
  };
}
```

File: src/hooks/useTelegramNotifications.ts

```
import { useInfiniteQuery } from '@tanstack/react-query';
import axios from 'axios';

// Types
export interface TelegramNotification {
  id: string;
  message: string;
  sent_at: string;
  status: 'pending' | 'sent' | 'delivered' | 'failed';
  metadata: Record<string, any>;
}

export interface NotificationsResponse {
  notifications: TelegramNotification[];
  nextCursor: string | null;
}

// Hook để lấy lịch sử thông báo Telegram
export function useTelegramNotifications() {
  const fetchNotifications = async ({ pageParam = '' }) => {
    const { data } = await axios.get<NotificationsResponse>(
      `/api/telegram/notifications?cursor=${pageParam}`
    );
    return data;
  };

  const query = useInfiniteQuery({
    queryKey: ['telegram', 'notifications'],
    queryFn: fetchNotifications,
    initialPageParam: '',
    getNextPageParam: (lastPage) => lastPage.nextCursor,
    staleTime: 1000 * 60 * 2, // 2 minutes
  });

  // Flatten pages results
  const notifications = query.data?.pages.flatMap(page =>
page.notifications) || [];
```

```
  return {
    notifications,
    isLoading: query.isLoading,
    error: query.error,
    fetchNextPage: query.fetchNextPage,
    hasNextPage: query.hasNextPage,
    isFetchingNextPage: query.isFetchingNextPage,
  };
}
```

```
### Prompt 10: Tạo React Components cho Telegram
```

Tạo chi tiết các React Components cho tính năng thông báo Telegram:

File: src/components/TelegramConnection.tsx

```
'use client';

import { useState } from 'react';
import { useTelegramConnection } from '@/hooks/useTelegramConnection';
import { Button } from '@/components/ui/button';
import { Card, CardContent, CardDescription, CardFooter, CardHeader,
CardTitle } from '@/components/ui/card';
import { Alert, AlertDescription, AlertTitle } from
'@/components/ui/alert';
import { QrCode, AlertCircle, RefreshCw, Check, X } from 'lucide-react';

export function TelegramConnection() {
  const {
    status,
    isConnected,
    isLoading,
    generateCode,
    isGeneratingCode,
    verificationData,
    disconnect,
    isDisconnecting,
  } = useTelegramConnection();

  const [showQRCode, setShowQRCode] = useState(false);

  // Xử lý kết nối
  const handleConnect = () => {
    generateCode();
  };

  // Xử lý ngắt kết nối
```

```
  const handleDisconnect = () => {
    if (confirm('Bạn có chắc muốn ngắt kết nối Telegram?')) {
      disconnect();
    }
  };

  // Hiển thị trạng thái kết nối
  const renderConnectionStatus = () => {
    if (isLoading) {
      return (
        <div className="flex items-center space-x-2">
          <RefreshCw className="animate-spin h-4 w-4" />
          <span>Đang tải...</span>
        </div>
      );
    }

    if (isConnected) {
      return (
        <div className="space-y-4">
          <div className="flex items-center space-x-2 text-green-600">
            <Check className="h-5 w-5" />
            <span className="font-medium">Đã kết nối với Telegram</span>
          </div>

          <div className="text-sm space-y-2">
            <p><span className="font-medium">Tài khoản:</span>
@{status?.username}</p>
            <p><span className="font-medium">Kết nối từ:</span> {new
Date(status?.connectedAt || '').toLocaleString('vi-VN')}</p>
          </div>

          <Button
            variant="destructive"
            size="sm"
            onClick={handleDisconnect}
            disabled={isDisconnecting}
          >
            {isDisconnecting ? <RefreshCw className="mr-2 h-4 w-4 animate-
spin" /> : <X className="mr-2 h-4 w-4" />}
            Ngắt kết nối
          </Button>
        </div>
      );
    }

    return (
      <div className="space-y-4">
        <div className="flex items-center space-x-2 text-yellow-600">
          <AlertCircle className="h-5 w-5" />
          <span className="font-medium">Chưa kết nối với Telegram</span>
        </div>

        <p className="text-sm text-muted-foreground">
```

```
          Kết nối với Telegram để nhận thông báo quan trọng về tài khoản,
giao dịch và cá cược.
        </p>

        <div className="flex flex-col space-y-3">
          <Button
            onClick={handleConnect}
            disabled={isGeneratingCode}
          >
            {isGeneratingCode ? <RefreshCw className="mr-2 h-4 w-4
animate-spin" /> : <QrCode className="mr-2 h-4 w-4" />}
            Kết nối với Telegram
          </Button>
        </div>
      </div>
    );
  };

  // Hiển thị hướng dẫn kết nối và QR code
  const renderConnectionInstructions = () => {
    if (!verificationData) return null;

    return (
      <Alert className="mt-4">
        <AlertTitle className="font-medium">Hướng dẫn kết nối
Telegram</AlertTitle>
        <AlertDescription className="mt-2">
          <ol className="list-decimal list-inside space-y-2 text-sm">
            <li>Mở ứng dụng Telegram trên điện thoại</li>
            <li>Tìm bot: @{verificationData.botUsername}</li>
            <li>Hoặc <a href={verificationData.deepLink} target="_blank"
rel="noopener noreferrer" className="text-primary underline">nhấn vào
đây</a> để mở bot trực tiếp</li>
            <li>Gửi lệnh: <code className="bg-muted px-1 py-0.5
rounded">/verify_{verificationData.verificationCode}</code></li>
          </ol>

          {showQRCode && (
            <div className="mt-4 flex justify-center">
              <img
                src={`https://api.qrserver.com/v1/create-qr-code/?
size=200x200&data=${encodeURIComponent(verificationData.deepLink)}`}
                alt="QR Code kết nối Telegram"
                className="border rounded"
              />
            </div>
          )}

          <Button
            variant="outline"
            size="sm"
            className="mt-3"
            onClick={() => setShowQRCode(!showQRCode)}
          >
```

```
              {showQRCode ? 'Ẩn QR code' : 'Hiển thị QR code'}
            </Button>
          </AlertDescription>
        </Alert>
      );
    };

    return (
      <Card className="w-full max-w-md mx-auto">
        <CardHeader>
          <CardTitle>Kết nối Telegram</CardTitle>
          <CardDescription>
            Nhận thông báo ngay lập tức về các sự kiện quan trọng qua
Telegram
          </CardDescription>
        </CardHeader>
        <CardContent>
          {renderConnectionStatus()}
          {renderConnectionInstructions()}
        </CardContent>
      </Card>
    );
}
```

File: src/components/TelegramSettings.tsx

```
'use client';

import { useTelegramConnection } from '@/hooks/useTelegramConnection';
import { Card, CardContent, CardDescription, CardFooter, CardHeader,
CardTitle } from '@/components/ui/card';
import { Switch } from '@/components/ui/switch';
import { Label } from '@/components/ui/label';
import { Button } from '@/components/ui/button';
import { RefreshCw, Bell, ShieldAlert, Wallet, Trophy, Info } from
'lucide-react';
import { useState, useEffect } from 'react';

export function TelegramSettings() {
  const { status, isConnected, isLoading, updateSettings,
isUpdatingSettings } = useTelegramConnection();

  const [settings, setSettings] = useState({
    account: true,
    payment: true,
    bet: true,
    system: true
  });

  // Load settings from status
  useEffect(() => {
    if (status?.settings) {
```

```
        setSettings(status.settings);
      }
    }, [status?.settings]);

    // Handle toggle change
    const handleToggleChange = (key: keyof typeof settings) => {
      const newSettings = { ...settings, [key]: !settings[key] };
      setSettings(newSettings);
    };

    // Save settings
    const handleSaveSettings = () => {
      updateSettings(settings);
    };

    if (!isConnected) {
      return (
        <Card className="w-full max-w-md mx-auto">
          <CardHeader>
            <CardTitle>Cài đặt thông báo Telegram</CardTitle>
            <CardDescription>
              Tùy chỉnh loại thông báo bạn muốn nhận qua Telegram
            </CardDescription>
          </CardHeader>
          <CardContent>
            <div className="flex items-center justify-center py-6 text-sm
text-muted-foreground">
              Vui lòng kết nối tài khoản Telegram trước
            </div>
          </CardContent>
        </Card>
      );
    }

    return (
      <Card className="w-full max-w-md mx-auto">
        <CardHeader>
          <CardTitle>Cài đặt thông báo Telegram</CardTitle>
          <CardDescription>
            Tùy chỉnh loại thông báo bạn muốn nhận qua Telegram
          </CardDescription>
        </CardHeader>
        <CardContent className="space-y-4">
          <div className="space-y-4">
            <div className="flex items-center justify-between">
              <div className="flex items-center space-x-2">
                <ShieldAlert className="h-4 w-4 text-blue-500" />
                <Label htmlFor="account" className="font-medium">Thông báo
tài khoản</Label>
              </div>
              <Switch
                id="account"
                checked={settings.account}
                onCheckedChange={() => handleToggleChange('account')}
```

```
              />
            </div>
            <p className="text-sm text-muted-foreground pl-6">
              Đăng nhập, thay đổi mật khẩu, cập nhật thông tin
            </p>
          </div>

          <div className="space-y-4">
            <div className="flex items-center justify-between">
              <div className="flex items-center space-x-2">
                <Wallet className="h-4 w-4 text-green-500" />
                <Label htmlFor="payment" className="font-medium">Thông báo
giao dịch</Label>
              </div>
              <Switch
                id="payment"
                checked={settings.payment}
                onCheckedChange={() => handleToggleChange('payment')}
              />
            </div>
            <p className="text-sm text-muted-foreground pl-6">
              Nạp tiền, rút tiền, thay đổi số dư
            </p>
          </div>

          <div className="space-y-4">
            <div className="flex items-center justify-between">
              <div className="flex items-center space-x-2">
                <Trophy className="h-4 w-4 text-yellow-500" />
                <Label htmlFor="bet" className="font-medium">Thông báo cá
cược</Label>
              </div>
              <Switch
                id="bet"
                checked={settings.bet}
                onCheckedChange={() => handleToggleChange('bet')}
              />
            </div>
            <p className="text-sm text-muted-foreground pl-6">
              Kết quả cá cược, thắng lớn, jackpot
            </p>
          </div>

          <div className="space-y-4">
            <div className="flex items-center justify-between">
              <div className="flex items-center space-x-2">
                <Info className="h-4 w-4 text-purple-500" />
                <Label htmlFor="system" className="font-medium">Thông báo hệ
thống</Label>
              </div>
              <Switch
                id="system"
                checked={settings.system}
                onCheckedChange={() => handleToggleChange('system')}
```

```
            />
          </div>
          <p className="text-sm text-muted-foreground pl-6">
            Bảo trì, cập nhật, khuyến mãi, thông báo quan trọng
          </p>
        </div>
      </CardContent>
      <CardFooter>
        <Button
          onClick={handleSaveSettings}
          disabled={isUpdatingSettings}
          className="w-full"
        >
          {isUpdatingSettings && <RefreshCw className="mr-2 h-4 w-4
animate-spin" />}
          Lưu cài đặt
        </Button>
      </CardFooter>
    </Card>
  );
}
```

```
### Prompt 11: Tích hợp Telegram vào hệ thống thông báo
```

Tạo code để tích hợp thông báo Telegram vào hệ thống thông báo hiện tại:

File: src/app/main/settings/notifications/page.tsx

```
import { Metadata } from 'next';
import { TelegramConnection } from '@/components/TelegramConnection';
import { TelegramSettings } from '@/components/TelegramSettings';
import { Tabs, TabsContent, TabsList, TabsTrigger } from
'@/components/ui/tabs';
import { Separator } from '@/components/ui/separator';

export const metadata: Metadata = {
  title: 'Cài đặt thông báo | VinBet',
  description: 'Quản lý cài đặt thông báo',
};

export default function NotificationsPage() {
  return (
    <div className="container mx-auto py-6 space-y-6">
      <div>
        <h1 className="text-2xl font-bold tracking-tight">Cài đặt thông
báo</h1>
        <p className="text-muted-foreground">
```

```
            Quản lý cách bạn nhận thông báo từ hệ thống
          </p>
        </div>

        <Separator />

        <Tabs defaultValue="app" className="w-full">
          <TabsList className="grid w-full grid-cols-2">
            <TabsTrigger value="app">Thông báo ứng dụng</TabsTrigger>
            <TabsTrigger value="telegram">Thông báo Telegram</TabsTrigger>
          </TabsList>

          <TabsContent value="app" className="space-y-6 py-4">
            {/* Existing app notification settings */}
            <h2 className="text-lg font-medium">Cài đặt thông báo ứng
dụng</h2>
            {/* App notification components go here */}
          </TabsContent>

          <TabsContent value="telegram" className="space-y-6 py-4">
            <h2 className="text-lg font-medium">Thông báo Telegram</h2>
            <p className="text-sm text-muted-foreground">
              Kết nối và quản lý thông báo qua Telegram để không bỏ lỡ các
thông tin quan trọng
            </p>

            <div className="grid gap-6 lg:grid-cols-2">
              <TelegramConnection />
              <TelegramSettings />
            </div>
          </TabsContent>
        </Tabs>
      </div>
    );
}
```

File: src/app/main/settings/notifications/history/page.tsx

```
'use client';

import { useState } from 'react';
import { useRouter } from 'next/navigation';
import { useTelegramNotifications } from
'@/hooks/useTelegramNotifications';
import { Button } from '@/components/ui/button';
import { Card } from '@/components/ui/card';
import { Tabs, TabsContent, TabsList, TabsTrigger } from
'@/components/ui/tabs';
import { Separator } from '@/components/ui/separator';
import { Badge } from '@/components/ui/badge';
import { CheckCircle2, XCircle, Clock, AlertCircle, Loader2 } from
'lucide-react';
```

```
export default function NotificationHistoryPage() {
  const router = useRouter();
  const { notifications, isLoading, fetchNextPage, hasNextPage,
isFetchingNextPage } = useTelegramNotifications();
  const [notificationType, setNotificationType] = useState<string>('all');

  // Filter notifications by type
  const filteredNotifications = notificationType === 'all'
    ? notifications
    : notifications.filter(n => n.metadata?.type === notificationType);

  // Status icon
  const getStatusIcon = (status: string) => {
    switch (status) {
      case 'sent':
      case 'delivered':
        return <CheckCircle2 className="h-4 w-4 text-green-500" />;
      case 'failed':
        return <XCircle className="h-4 w-4 text-red-500" />;
      case 'pending':
        return <Clock className="h-4 w-4 text-yellow-500" />;
      default:
        return <AlertCircle className="h-4 w-4 text-gray-500" />;
    }
  };

  // Notification type badge
  const getTypeBadge = (type: string) => {
    switch (type) {
      case 'account':
        return <Badge variant="outline" className="bg-blue-50 text-blue-
700 border-blue-200">Tài khoản</Badge>;
      case 'payment':
        return <Badge variant="outline" className="bg-green-50 text-green-
700 border-green-200">Giao dịch</Badge>;
      case 'bet':
        return <Badge variant="outline" className="bg-yellow-50 text-
yellow-700 border-yellow-200">Cá cược</Badge>;
      case 'system':
        return <Badge variant="outline" className="bg-purple-50 text-
purple-700 border-purple-200">Hệ thống</Badge>;
      default:
        return <Badge variant="outline">Khác</Badge>;
    }
  };

  return (
    <div className="container mx-auto py-6 space-y-6">
      <div className="flex items-center justify-between">
        <div>
          <h1 className="text-2xl font-bold tracking-tight">Lịch sử thông
báo</h1>
          <p className="text-muted-foreground">
```

```
              Xem lại các thông báo đã gửi qua Telegram
            </p>
          </div>
          <Button variant="outline" onClick={() => router.back()}>
            Quay lại
          </Button>
        </div>

        <Separator />

        <Tabs
          defaultValue="all"
          value={notificationType}
          onValueChange={setNotificationType}
          className="w-full"
        >
          <TabsList className="grid w-full grid-cols-5">
            <TabsTrigger value="all">Tất cả</TabsTrigger>
            <TabsTrigger value="account">Tài khoản</TabsTrigger>
            <TabsTrigger value="payment">Giao dịch</TabsTrigger>
            <TabsTrigger value="bet">Cá cược</TabsTrigger>
            <TabsTrigger value="system">Hệ thống</TabsTrigger>
          </TabsList>
        </Tabs>

        <div className="space-y-4">
          {isLoading ? (
            <div className="flex justify-center py-12">
              <Loader2 className="h-8 w-8 animate-spin text-primary" />
            </div>
          ) : filteredNotifications.length === 0 ? (
            <Card className="p-6 text-center text-muted-foreground">
              Không có thông báo nào trong mục này
            </Card>
          ) : (
            <>
              {filteredNotifications.map((notification) => (
                <Card key={notification.id} className="p-4 hover:bg-
accent/30 transition-colors">
                  <div className="flex items-start gap-3">
                    <div className="mt-1">
                      {getStatusIcon(notification.status)}
                    </div>
                    <div className="flex-1 space-y-1.5">
                      <div className="flex items-center justify-between">
                        <div className="font-medium">
                          {getTypeBadge(notification.metadata?.type)}
                        </div>
                        <div className="text-xs text-muted-foreground">
                          {new
Date(notification.sent_at).toLocaleString('vi-VN')}
                        </div>
                      </div>
                      <p className="text-sm whitespace-pre-line">
```

```
                      {notification.message}
                    </p>
                  </div>
                </div>
              </Card>
            ))}

            {hasNextPage && (
              <div className="flex justify-center pt-4">
                <Button
                  variant="outline"
                  onClick={() => fetchNextPage()}
                  disabled={isFetchingNextPage}
                >
                  {isFetchingNextPage ? (
                    <><Loader2 className="mr-2 h-4 w-4 animate-spin" />
Đang tải...</>
                  ) : (
                    'Xem thêm'
                  )}
                </Button>
              </div>
            )}
          </>
        )}
      </div>
    </div>
  );
}
```

```
## IV. Tạo Telegram Bot và các thông báo mẫu

### Prompt 12: Tạo Telegram Bot và mẫu thông báo
```

Tạo hướng dẫn chi tiết về cách tạo Telegram Bot và các mẫu thông báo:

Hướng dẫn tạo Telegram Bot

1. **Mở BotFather trên Telegram**
   - Tìm @BotFather trên Telegram hoặc truy cập t.me/BotFather
   - Gửi lệnh /start để bắt đầu
2. **Tạo bot mới**
   - Gửi lệnh /newbot
   - Nhập tên hiển thị cho bot (ví dụ: VinBet Notifications)
   - Nhập username cho bot (phải kết thúc bằng 'bot', ví dụ: vinbet_notif_bot)

3. **Lưu API Token**
    - BotFather sẽ trả về một token dạng: 123456789:ABCDefGhIJKlmNoPQRsTUVwxyZ
    - Lưu token này vào biến môi trường TELEGRAM_BOT_TOKEN trong file .env
4. **Thiết lập các lệnh cho bot**
    - Gửi lệnh /setcommands đến BotFather
    - Chọn bot của bạn
    - Dán danh sách lệnh sau:

```
start - Bắt đầu sử dụng bot
help - Xem hướng dẫn sử dụng
status - Kiểm tra trạng thái kết nối
settings - Xem cài đặt thông báo
```

5. **Tùy chỉnh bot**
    - Gửi lệnh /setdescription để cài đặt mô tả cho bot
    - Gửi lệnh /setabouttext để cài đặt thông tin giới thiệu
    - Gửi lệnh /setuserpic để cài đặt ảnh đại diện cho bot

Mẫu thông báo Telegram (sử dụng Markdown)

1. **Thông báo đăng nhập**

```
🔐 *Đăng nhập mới*

Tài khoản của bạn vừa được đăng nhập từ một thiết bị mới.

📱 Thiết bị: {device}
📍 Vị trí: {location}
🕒 Thời gian: {time}

Nếu không phải bạn, vui lòng thay đổi mật khẩu ngay lập tức!
```

2. **Thông báo nạp tiền thành công**

```
💰 *Nạp tiền thành công*

✅ Giao dịch nạp tiền của bạn đã được duyệt.

💸 Số tiền: {amount} VND
🏦 Phương thức: {method}
🕒 Thời gian: {time}
🔢 Mã giao dịch: {transaction_id}

Số dư hiện tại của bạn là {balance} VND.
```

3. **Thông báo rút tiền thành công**

```
💸 *Rút tiền thành công*

✅ Yêu cầu rút tiền của bạn đã được xử lý thành công.

💰 Số tiền: {amount} VND
🏦 Đến: {bank_info}
🕒 Thời gian: {time}
🔢 Mã giao dịch: {transaction_id}

Số dư hiện tại của bạn là {balance} VND.
```

4. **Thông báo thắng cược lớn**

```
🎉 *Chúc mừng bạn thắng lớn!*

🎮 Game: {game_name}
💰 Tiền cược: {bet_amount} VND
🏆 Tiền thắng: {win_amount} VND
✨ Tỷ lệ thắng: x{multiplier}
🕒 Thời gian: {time}

Số dư hiện tại của bạn là {balance} VND.
```

5. **Thông báo bảo trì hệ thống**

```
⚠ *Thông báo bảo trì hệ thống*

🔧 VinBet sẽ tiến hành bảo trì hệ thống từ:
   {start_time} đến {end_time}

⏱ Thời gian dự kiến: {duration} phút

Trong thời gian này, một số tính năng có thể không khả dụng. Chúng tôi sẽ
thông báo cho bạn khi quá trình bảo trì hoàn tất.

Xin lỗi vì sự bất tiện này và cảm ơn sự thông cảm của bạn!
```

6. **Thông báo khuyến mãi**

```
🎁 *Khuyến mãi đặc biệt*

📣 {promotion_title}
```

```
{promotion_description}

⏰ Thời hạn: {end_date}
🔗 Chi tiết: {link}

Đừng bỏ lỡ cơ hội này!
```

Các mẫu message cho Telegram Bot

1. **Tin nhắn chào mừng khi người dùng gửi /start**

```
Chào mừng bạn đến với VinBet Notifications! 👋

Bot này sẽ gửi cho bạn các thông báo quan trọng từ tài khoản VinBet của
bạn.

Để kết nối với tài khoản VinBet:
1. Đăng nhập vào tài khoản VinBet
2. Vào Cài đặt > Thông báo > Telegram
3. Nhấn "Kết nối với Telegram"
4. Sao chép mã xác thực và gửi lệnh /verify_CODE cho bot này

Gửi /help để xem danh sách các lệnh có sẵn.
```

2. **Tin nhắn trợ giúp khi người dùng gửi /help**

```
📚 Danh sách lệnh:

/start - Bắt đầu sử dụng bot
/help - Xem hướng dẫn này
/status - Kiểm tra trạng thái kết nối
/settings - Xem cài đặt thông báo hiện tại

Nếu bạn cần hỗ trợ thêm, vui lòng liên hệ với đội ngũ hỗ trợ của VinBet.
```

3. **Tin nhắn xác nhận khi người dùng kết nối thành công**

```
✅ Kết nối thành công!

Tài khoản Telegram của bạn đã được liên kết với tài khoản VinBet.

Bạn sẽ nhận được thông báo về:
- Đăng nhập và bảo mật tài khoản
- Giao dịch nạp/rút tiền
- Kết quả cá cược và thắng lớn
- Thông báo hệ thống và khuyến mãi
```

```
Bạn có thể thay đổi cài đặt thông báo trong trang cài đặt VinBet hoặc gửi
lệnh /settings.
```

```
Với kế hoạch triển khai chi tiết này, Chat Bot AI sẽ có đủ thông tin cụ
thể để thực hiện từng bước một cách chính xác. Mỗi prompt đã được thiết kế
với:

1. Ngữ cảnh cụ thể dựa trên hệ thống hiện tại
2. Code hoàn chỉnh với comments và chi tiết triển khai
3. Mẫu rõ ràng cho từng component và function
4. Hướng dẫn chi tiết từng bước
5. Xử lý edge cases và error handling

Liệu bạn cần thêm chi tiết nào khác cho kế hoạch này?
```