Ga cikakken rubutun main.py da aka gyara:

```
import os
import logging
import requests
import json
from telegram import Update, InlineKeyboardButton,
InlineKeyboardMarkup
from telegram.constants import ChatAction
from telegram.ext import (
    Application,
    CommandHandler,
    MessageHandler,
    CallbackQueryHandler,
    filters,
    ContextTypes
)
from dotenv import load_dotenv
import urllib.parse

# Load environment variables from .env file
load_dotenv()

TELEGRAM_BOT_TOKEN = os.getenv("BOT_TOKEN")
if not TELEGRAM_BOT_TOKEN:
    raise ValueError("⚠️ BOT_TOKEN bai da kyau ko bai samu ba daga
.env file. Tabbatar ka saka BOT_TOKEN=... a cikin .env")

# SAITA ID DIN ADMIN ANAN!
# Ka canja '0000000000' zuwa naka Telegram User ID na gaske.
# Zaka iya samun ID din ka ta hanyar aika sako ga @userinfobot a
Telegram.
ADMIN_USER_ID = 0000000000 # <-- CANJA WANNAN ZUWA NAKA ID NA
TELEGRAM!

COINGECKO_API_URL = "https://api.coingecko.com/api/v3/simple/price"
ALERTS_FILE = "alerts.json"
SETUP_GUIDE_LINK = "https://t.me/c/2544548450/3"

def load_alerts():
    """Loads alerts from the JSON file."""
    if os.path.exists(ALERTS_FILE):
        with open(ALERTS_FILE, "r") as f:
            try:
                return json.load(f)
            except json.JSONDecodeError:
                logger.warning("alerts.json is empty or malformed,
initializing with empty dictionary.")
                return {}
```

```
    return {}

def save_alerts(data):
    """Saves alerts to the JSON file."""
    with open(ALERTS_FILE, "w") as f:
        json.dump(data, f, indent=2)

price_alerts = load_alerts()

logging.basicConfig(format='%(asctime)s - %(name)s - %(levelname)s -
%(message)s', level=logging.INFO)
logger = logging.getLogger(__name__)

async def send_main_menu(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    """Sends or edits the main menu message."""
    menu_text = (
        "🤖 *Crypto Price Bot Menu*\n"
        "Zaɓi ɗaya daga cikin abubuwa masu zuwa:"
    )
    keyboard = [
        [InlineKeyboardButton("📚 Setup Guide",
url=SETUP_GUIDE_LINK)],
        [InlineKeyboardButton("📈 Duba Farashin Cryptocurrency",
callback_data="show_price_info")],
        [InlineKeyboardButton("🔔 Saita Faɗakarwar Farashi",
callback_data="show_alert_info")],
        [InlineKeyboardButton("📋 Duba Faɗakarwarka",
callback_data="my_alerts_button")],
        [InlineKeyboardButton("📊 Chart na Cryptocurrency",
callback_data="show_chart_info")],
    ]
    reply_markup = InlineKeyboardMarkup(keyboard)

    if update.callback_query:
        try:
            await update.callback_query.edit_message_text(menu_text,
reply_markup=reply_markup, parse_mode="Markdown")
        except Exception as e:
            logger.warning(f"Failed to edit message for main menu:
{e}. Sending new message.")
            await update.callback_query.message.reply_text(menu_text,
reply_markup=reply_markup, parse_mode="Markdown")
    elif update.message:
        await update.message.reply_text(menu_text,
reply_markup=reply_markup, parse_mode="Markdown")
```

```
async def show_price_info(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    """Provides information on how to use the /price command."""
    query = update.callback_query
    await query.answer()
    info_text = (
        "📈 *Duba Farashin Coin*\n\n"
        "Don duba farashin coin, yi amfani da umarnin `/price` sannan
ka rubuta sunan coin din.\n\n"
        "Misali: `/price bitcoin` ko `/price ethereum`"
    )
    keyboard = [[InlineKeyboardButton("🔙 Komawa Menu",
callback_data="back_to_main_menu")]]
    reply_markup = InlineKeyboardMarkup(keyboard)
    await query.edit_message_text(info_text,
reply_markup=reply_markup, parse_mode="Markdown")

async def show_alert_info(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    """Provides information on how to use the /alert command."""
    query = update.callback_query
    await query.answer()
    info_text = (
        "🚨 *Saita Faɗakarwar Farashi*\n\n"
        "Don saita faɗakarwa, yi amfani da umarnin `/alert` sannan ka
rubuta sunan coin, adadin, da kuma 'up' ko 'down'.\n\n"
        "Misali: `/alert ethereum 3000 up` (Idan farashin Ethereum ya
kai $3000 ko sama, ka fada min)\n"
        "Misali: `/alert bitcoin 25000 down` (Idan farashin Bitcoin ya
faɗi kasa da $25000, ka fada min)"
    )
    keyboard = [[InlineKeyboardButton("🔙 Komawa Menu",
callback_data="back_to_main_menu")]]
    reply_markup = InlineKeyboardMarkup(keyboard)
    await query.edit_message_text(info_text,
reply_markup=reply_markup, parse_mode="Markdown")

async def show_chart_info(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    """Provides information on how to use the /chart command."""
    query = update.callback_query
    await query.answer()
    info_text = (
        "📊 *Duba Chart na Cryptocurrency*\n\n"
        "Don ganin chart na coin, yi amfani da umarnin `/chart` sannan
ka rubuta sunan coin din.\n\n"
        "Misali: `/chart bitcoin` ko `/chart ethereum`"
    )
```

```
    keyboard = [[InlineKeyboardButton("🔙 Komawa Menu",
callback_data="back_to_main_menu")]]
    reply_markup = InlineKeyboardMarkup(keyboard)
    await query.edit_message_text(info_text,
reply_markup=reply_markup, parse_mode="Markdown")


async def start_command(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    """Handles the /start command and provides the initial welcome
message with main menu buttons."""
    await send_main_menu(update, context)


async def help_command(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    """Provides help information."""
    help_message = (
        "📚 *Taimako da Bayani*\n\n"
        "Ga jerin umarnoni da zaka iya amfani dasu:\n"
        "`/start` - 🤖 Fara amfani da bot (Zai nuna babban menu)\n"
        "`/menu` - 📋 Nuna babban menu\n"
        "`/price <coin_name>` - 📈 Duba farashin coin (misali: `/price
bitcoin`)\n"
        "`/alert <coin_name> <price> <up/down>` - 🔔 Saita faɗakarwa
(misali: `/alert ethereum 3000 up`)\n"
        "`/myalerts` - 📋 Duba faɗakarwar da aka saita\n"
        "`/cancelalert <coin_name>` - ❌ Soke faɗakarwa (misali:
`/cancelalert bitcoin`)\n"
        "`/chart <coin_name>` - 📊 Duba chart na coin (misali: `/chart
bitcoin`)"
    )
    keyboard = [[InlineKeyboardButton("🔙 Komawa Menu",
callback_data="back_to_main_menu")]]
    reply_markup = InlineKeyboardMarkup(keyboard)

    if update.message:
        await update.message.reply_text(help_message,
reply_markup=reply_markup, parse_mode="Markdown")
    elif update.callback_query:
        await update.callback_query.answer()
        await update.callback_query.edit_message_text(help_message,
reply_markup=reply_markup, parse_mode="Markdown")

async def get_price(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    """Fetches and displays the price of a given cryptocurrency."""
    if not context.args:
```

```
        keyboard = [[InlineKeyboardButton("🔙 Komawa Menu",
callback_data="back_to_main_menu")]]
        reply_markup = InlineKeyboardMarkup(keyboard)
        sent_message = await update.message.reply_text(
            "Don Allah ka bayar da sunan coin. Misali: `/price
bitcoin`",
            reply_markup=reply_markup,
            parse_mode="Markdown"
        )
        context.user_data['last_editable_message_id'] =
sent_message.message_id
        context.user_data['last_editable_chat_id'] =
sent_message.chat_id
        return

    await update.message.chat.send_action(action=ChatAction.TYPING)

    coin_name = " ".join(context.args).lower()
    price_message = ""
    try:
        search_url =
f"https://api.coingecko.com/api/v3/search?query={coin_name}"
        search_response = requests.get(search_url).json()
        coin_id = None
        for coin in search_response.get('coins', []):
            if coin['symbol'].lower() == coin_name or
coin['name'].lower() == coin_name:
                coin_id = coin['id']
                break
        if not coin_id:
            price_message = f"Ba a sami coin din '{coin_name}' ba. Don
Allah tabbatar da sunan coin daidai ne."
        else:
            params = {"ids": coin_id, "vs_currencies": "usd"}
            response = requests.get(COINGECKO_API_URL,
params=params).json()
            price = response.get(coin_id, {}).get('usd')
            if price is not None:
                price_message = f"Farashin {coin_name.capitalize()} a
yanzu shine: ${price:,.2f}"
            else:
                price_message = f"Ba a sami farashin {coin_name} ba a
yanzu. Gwada anjima."
    except Exception as e:
        logger.error(f"Kuskure yayin samun farashi: {e}")
        price_message = "An samu kuskure yayin kokarin samun farashin.
Don Allah gwada anjima."
```

```
    keyboard = [[InlineKeyboardButton("🔙 Komawa Menu",
callback_data="back_to_main_menu")]]
    reply_markup = InlineKeyboardMarkup(keyboard)

    sent_message = await update.message.reply_text(price_message,
reply_markup=reply_markup, parse_mode="Markdown")
    context.user_data['last_editable_message_id'] =
sent_message.message_id
    context.user_data['last_editable_chat_id'] = sent_message.chat_id


async def set_alert(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    """Sets a price alert for a cryptocurrency."""
    alert_message = ""
    if len(context.args) < 3:
        alert_message = "Don Allah ka bayar da sunan coin, farashi, da
kuma shugabanci (up/down). Misali: `/alert ethereum 3000 up`"
    else:
        coin_name = context.args[0].lower()
        try:
            target_price = float(context.args[1])
        except ValueError:
            alert_message = "Farashin da ka bayar ba lamba ba ce.
Misali: `/alert ethereum 3000 up`"
        else:
            direction = context.args[2].lower()
            if direction not in ['up', 'down']:
                alert_message = "Shugabancin dole ne ya zama 'up' ko
'down'. Misali: `/alert ethereum 3000 up`"
            else:
                chat_id = update.effective_chat.id
                try:
                    search_url =
f"https://api.coingecko.com/api/v3/search?query={coin_name}"
                    search_response = requests.get(search_url).json()
                    coin_id = None
                    for coin in search_response.get('coins', []):
                        if coin['symbol'].lower() == coin_name or
coin['name'].lower() == coin_name:
                            coin_id = coin['id']
                            break
                    if not coin_id:
                        alert_message = f"Ba a sami coin din
'{coin_name}' ba. Don Allah tabbatar da sunan coin daidai ne."
                    else:
                        if chat_id not in price_alerts:
                            price_alerts[chat_id] = {}
```

```
                        price_alerts[chat_id][coin_id] = {
                            'target_price': target_price,
                            'direction': direction,
                            'original_coin_name': coin_name
                        }
                        save_alerts(price_alerts)
                        alert_message = f"An saita faɗakarwa don
{coin_name.capitalize()}: lokacin da farashin ya kai
${target_price:,.2f} ({direction})."
                except Exception as e:
                    logger.error(f"Kuskure yayin saita faɗakarwa: {e}")
                    alert_message = "An samu kuskure yayin saita
faɗakarwa. Don Allah gwada anjima."

    keyboard = [[InlineKeyboardButton("🔙 Komawa Menu",
callback_data="back_to_main_menu")]]
    reply_markup = InlineKeyboardMarkup(keyboard)

    sent_message = await update.message.reply_text(alert_message,
reply_markup=reply_markup, parse_mode="Markdown")
    context.user_data['last_editable_message_id'] =
sent_message.message_id
    context.user_data['last_editable_chat_id'] = sent_message.chat_id


async def my_alerts_command(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    """Displays the list of active price alerts for the user."""
    chat_id = update.effective_chat.id
    alerts = price_alerts.get(chat_id, {})
    message_text = ""
    if not alerts:
        message_text = "Ba ku da faɗakarwar farashi a yanzu."
    else:
        message_text = "📋 *Faɗakarwarku na farashin:*\n"
        for coin_id, info in alerts.items():
            name = info.get('original_coin_name', coin_id)
            message_text += f"- {name.capitalize()}:
${info['target_price']:,.2f} ({info['direction']})\n"

    keyboard = [[InlineKeyboardButton("🔙 Komawa Menu",
callback_data="back_to_main_menu")]]
    reply_markup = InlineKeyboardMarkup(keyboard)

    if update.callback_query:
        await update.callback_query.answer()
        await update.callback_query.edit_message_text(message_text,
reply_markup=reply_markup, parse_mode="Markdown")
```

```
    else: # For direct command usage
        sent_message = await update.message.reply_text(message_text,
reply_markup=reply_markup, parse_mode="Markdown")
        context.user_data['last_editable_message_id'] =
sent_message.message_id
        context.user_data['last_editable_chat_id'] =
sent_message.chat_id


async def cancel_alert(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    """Cancels a specific price alert."""
    cancel_message = ""
    if not context.args:
        cancel_message = "Don Allah ka bayar da sunan coin da zaka
soke faɗakarwarsa. Misali: `/cancelalert bitcoin`"
    else:
        coin_name = " ".join(context.args).lower()
        chat_id = update.effective_chat.id
        alerts = price_alerts.get(chat_id, {})
        to_remove = None
        for coin_id, info in alerts.items():
            if info.get('original_coin_name', coin_id).lower() ==
coin_name:
                to_remove = coin_id
                break

        if to_remove:
            del alerts[to_remove]
            if not alerts:
                del price_alerts[chat_id]
            save_alerts(price_alerts)
            cancel_message = f"An soke faɗakarwa don
{coin_name.capitalize()}."
        else:
            cancel_message = f"Ba a sami faɗakarwa don '{coin_name}'
ba."

    keyboard = [[InlineKeyboardButton("🔙 Komawa Menu",
callback_data="back_to_main_menu")]]
    reply_markup = InlineKeyboardMarkup(keyboard)

    sent_message = await update.message.reply_text(cancel_message,
reply_markup=reply_markup, parse_mode="Markdown")
    context.user_data['last_editable_message_id'] =
sent_message.message_id
    context.user_data['last_editable_chat_id'] = sent_message.chat_id
```

```
async def send_price_chart(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    if not context.args:
        keyboard = [[InlineKeyboardButton("🔙 Komawa Menu",
callback_data="back_to_main_menu")]]
        reply_markup = InlineKeyboardMarkup(keyboard)
        sent_message = await update.message.reply_text(
            "Don Allah ka bayar da sunan coin don ganin chart. Misali:
`/chart bitcoin`",
            reply_markup=reply_markup,
            parse_mode="Markdown"
        )
        context.user_data['last_editable_message_id'] =
sent_message.message_id
        context.user_data['last_editable_chat_id'] =
sent_message.chat_id
        return

    await
update.message.chat.send_action(action=ChatAction.UPLOAD_PHOTO)

    coin_name = " ".join(context.args).lower()

    try:
        search_url =
f"https://api.coingecko.com/api/v3/search?query={coin_name}"
        search_response = requests.get(search_url).json()
        coin_id = None
        for coin in search_response.get('coins', []):
            if coin['symbol'].lower() == coin_name or
coin['name'].lower() == coin_name:
                coin_id = coin['id']
                break
        if not coin_id:
            chart_message = f"Ba a sami coin din '{coin_name}' ba. Don
Allah tabbatar da sunan coin daidai ne."
            keyboard = [[InlineKeyboardButton("🔙 Komawa Menu",
callback_data="back_to_main_menu")]]
            reply_markup = InlineKeyboardMarkup(keyboard)
            sent_message = await
update.message.reply_text(chart_message, reply_markup=reply_markup,
parse_mode="Markdown")
            context.user_data['last_editable_message_id'] =
sent_message.message_id
            context.user_data['last_editable_chat_id'] =
sent_message.chat_id
            return
```

```
        ohlc_url =
f"https://api.coingecko.com/api/v3/coins/{coin_id}/ohlc?vs_currency=us
d&days=3"
        ohlc_response = requests.get(ohlc_url).json()

        if not ohlc_response:
            chart_message = f"Ba a sami bayanan chart na '{coin_name}'
ba a yanzu."
            keyboard = [[InlineKeyboardButton("🔙 Komawa Menu",
callback_data="back_to_main_menu")]]
            reply_markup = InlineKeyboardMarkup(keyboard)
            sent_message = await
update.message.reply_text(chart_message, reply_markup=reply_markup,
parse_mode="Markdown")
            context.user_data['last_editable_message_id'] =
sent_message.message_id
            context.user_data['last_editable_chat_id'] =
sent_message.chat_id
            return

        labels = [data[0] for data in ohlc_response]
        open_data = [data[1] for data in ohlc_response]
        high_data = [data[2] for data in ohlc_response]
        low_data = [data[3] for data in ohlc_response]
        close_data = [data[4] for data in ohlc_response]

        candlestick_data = []
        for i in range(len(ohlc_response)):
            candlestick_data.append({
                "x": labels[i],
                "o": open_data[i],
                "h": high_data[i],
                "l": low_data[i],
                "c": close_data[i]
            })

        chart_config = {
            "type": "candlestick",
            "data": {
                "datasets": [{
                    "label": f"{coin_name.capitalize()} Price",
                    "data": candlestick_data,
                    "backgroundColor": "rgba(75, 192, 192, 0.2)",
                    "borderColor": "rgba(75, 192, 192, 1)",
                    "borderWidth": 1
                }]
            },
```

```
            "options": {
                "responsive": True,
                "maintainAspectRatio": False,
                "scales": {
                    "x": {
                        "type": "time",
                        "time": {
                            "unit": "day",
                            "displayFormats": {
                                "day": "MMM D"
                            }
                        },
                        "title": {
                            "display": True,
                            "text": "Kwanaki"
                        }
                    },
                    "y": {
                        "beginAtZero": False,
                        "title": {
                            "display": True,
                            "text": "Farashi (USD)"
                        }
                    }
                },
                "plugins": {
                    "title": {
                        "display": True,
                        "text": f"{coin_name.capitalize()} Candlestick
Chart (Kwana 3)"
                    }
                }
            }
        }

        quickchart_base_url = "https://quickchart.io/chart"
        encoded_chart_config =
urllib.parse.quote_plus(json.dumps(chart_config))
        quickchart_url =
f"{quickchart_base_url}?width=800&height=400&c={encoded_chart_config}"

        logger.info(f"Generated QuickChart URL:
{quickchart_url[:200]}...")

        keyboard = [[InlineKeyboardButton("🔙 Komawa Menu",
callback_data="back_to_main_menu")]]
        reply_markup = InlineKeyboardMarkup(keyboard)
```

```
        sent_message = await update.message.reply_photo(
            photo=quickchart_url,
            caption=f"Candlestick Chart na {coin_name.capitalize()}
(Kwana 3)",
            reply_markup=reply_markup
        )
        context.user_data['last_editable_message_id'] =
sent_message.message_id
        context.user_data['last_editable_chat_id'] =
sent_message.chat_id

    except Exception as e:
        logger.error(f"Kuskure yayin samun ko turawa chart: {e}")
        keyboard = [[InlineKeyboardButton("🔙 Komawa Menu",
callback_data="back_to_main_menu")]]
        reply_markup = InlineKeyboardMarkup(keyboard)
        sent_message = await update.message.reply_text("An samu
kuskure yayin kokarin samun chart. Don Allah gwada anjima.",
reply_markup=reply_markup, parse_mode="Markdown")
        context.user_data['last_editable_message_id'] =
sent_message.message_id
        context.user_data['last_editable_chat_id'] =
sent_message.chat_id


async def check_alerts(context: ContextTypes.DEFAULT_TYPE):
    """Periodically checks current prices against set alerts and
notifies users."""
    for chat_id, alerts in list(price_alerts.items()):
        for coin_id, info in list(alerts.items()):
            try:
                response = requests.get(COINGECKO_API_URL,
params={"ids": coin_id, "vs_currencies": "usd"}).json()
                current_price = response.get(coin_id, {}).get('usd')
                if current_price is None:
                    continue

                target_price = info['target_price']
                direction = info['direction']
                name = info.get('original_coin_name', coin_id)

                alert_triggered = False
                if direction == 'up' and current_price >=
target_price:
                    alert_triggered = True
                elif direction == 'down' and current_price <=
target_price:
                    alert_triggered = True
```

```
                if alert_triggered:
                    msg = f"📣 Faɗakarwa! Farashin {name.capitalize()}
ya kai ${current_price:,.2f}, wanda ya kai ko ya wuce
${target_price:,.2f} da ka saita."
                    await context.bot.send_message(chat_id=chat_id,
text=msg, parse_mode="Markdown")
                    del alerts[coin_id]
            except Exception as e:
                logger.error(f"Kuskure yayin duba faɗakarwa don
{coin_id}: {e}")

        if not alerts:
            del price_alerts[chat_id]
    save_alerts(price_alerts)

async def unknown(update: Update, context: ContextTypes.DEFAULT_TYPE):
    """Handles unknown commands."""
    keyboard = [[InlineKeyboardButton("🔙 Komawa Menu",
callback_data="back_to_main_menu")]]
    reply_markup = InlineKeyboardMarkup(keyboard)
    sent_message = await update.message.reply_text(
        "Ban gane wannan umarnin ba. Don Allah gwada umarni kamar
`/menu` ko `/price bitcoin`.",
        reply_markup=reply_markup,
        parse_mode="Markdown"
    )
    context.user_data['last_editable_message_id'] =
sent_message.message_id
    context.user_data['last_editable_chat_id'] = sent_message.chat_id


async def button_handler(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    """Handles all callback queries from inline keyboard buttons."""
    query = update.callback_query
    await query.answer()

    if query.data == "back_to_main_menu":
        try:
            await query.edit_message_text(
                "🤖 *Crypto Price Bot Menu*\nZaɓi ɗaya daga cikin
abubuwa masu zuwa:",
                reply_markup=InlineKeyboardMarkup([
                    [InlineKeyboardButton("📚 Setup Guide",
url=SETUP_GUIDE_LINK)],
                    [InlineKeyboardButton("📈 Duba Farashin
Cryptocurrency", callback_data="show_price_info")],
```

```
                    [InlineKeyboardButton("🔔 Saita Faɗakarwar
Farashi", callback_data="show_alert_info")],
                    [InlineKeyboardButton("📋 Duba Faɗakarwarka",
callback_data="my_alerts_button")],
                    [InlineKeyboardButton("📊 Chart na
Cryptocurrency", callback_data="show_chart_info")],
                ]),
                parse_mode="Markdown"
            )
            if 'last_editable_message_id' in context.user_data:
                del context.user_data['last_editable_message_id']
            if 'last_editable_chat_id' in context.user_data:
                del context.user_data['last_editable_chat_id']
        except Exception as e:
            logger.warning(f"Failed to edit callback query message on
back_to_main_menu: {e}")
            last_msg_id =
context.user_data.get('last_editable_message_id')
            last_chat_id =
context.user_data.get('last_editable_chat_id')
            if last_msg_id and last_chat_id:
                try:
                    await context.bot.edit_message_text(
                        chat_id=last_chat_id,
                        message_id=last_msg_id,
                        text="🤖 *Crypto Price Bot Menu*\nZaɓi ɗaya
daga cikin abubuwa masu zuwa:",
                        reply_markup=InlineKeyboardMarkup([
                            [InlineKeyboardButton("📚 Setup Guide",
url=SETUP_GUIDE_LINK)],
                            [InlineKeyboardButton("📈 Duba Farashin
Cryptocurrency", callback_data="show_price_info")],
                            [InlineKeyboardButton("🔔 Saita Faɗakarwar
Farashi", callback_data="show_alert_info")],
                            [InlineKeyboardButton("📋 Duba
Faɗakarwarka", callback_data="my_alerts_button")],
                            [InlineKeyboardButton("📊 Chart na
Cryptocurrency", callback_data="show_chart_info")],
                        ]),
                        parse_mode="Markdown"
                    )
                    if 'last_editable_message_id' in
context.user_data:
                        del
context.user_data['last_editable_message_id']
                    if 'last_editable_chat_id' in context.user_data:
                        del context.user_data['last_editable_chat_id']
                except Exception as e:
```

```
                    logger.warning(f"Failed to edit
last_editable_message_id on back_to_main_menu: {e}. Sending new menu
message.")
                    await send_main_menu(update, context)
            else:
                await send_main_menu(update, context)

    elif query.data == "show_price_info":
        await show_price_info(update, context)
    elif query.data == "show_alert_info":
        await show_alert_info(update, context)
    elif query.data == "my_alerts_button":
        await my_alerts_command(update, context)
    elif query.data == "show_chart_info":
        await show_chart_info(update, context)
    elif query.data == "guide":
        await help_command(update, context)

# === SABON ADMIN COMMAND ===
async def admin_stats(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    """
    Shows statistics about bot usage (number of users, number of
alerts).
    Only accessible by the ADMIN_USER_ID.
    """
    # Duba ko mai amfani shine ADMIN_USER_ID
    if update.effective_user.id != ADMIN_USER_ID:
        await update.message.reply_text("Ba ka da izinin yin wannan
umarnin.")
        return

    num_users = len(price_alerts) # Kowane key a price_alerts yana
wakiltar chat_id (mai amfani daban)

    total_alerts = 0
    for alerts_for_user in price_alerts.values():
        total_alerts += len(alerts_for_user) # Kidaya adadin alerts ga
kowane mai amfani

    stats_message = (
        "*Kididdigar Bot:*\n"
        f"Jimillar Masu Amfani: `{num_users}`\n"
        f"Jimillar Faɗakarwar Farashi: `{total_alerts}`"
    )

    keyboard = [[InlineKeyboardButton("🔙 Komawa Menu",
callback_data="back_to_main_menu")]]
```

```
    reply_markup = InlineKeyboardMarkup(keyboard)

    sent_message = await update.message.reply_text(stats_message,
reply_markup=reply_markup, parse_mode="Markdown")
    context.user_data['last_editable_message_id'] =
sent_message.message_id
    context.user_data['last_editable_chat_id'] = sent_message.chat_id


def main():
    """Starts the bot."""
    app = Application.builder().token(TELEGRAM_BOT_TOKEN).build()

    # Command Handlers
    app.add_handler(CommandHandler("start", start_command))
    app.add_handler(CommandHandler("help", help_command))
    app.add_handler(CommandHandler("menu", send_main_menu))
    app.add_handler(CommandHandler("price", get_price))
    app.add_handler(CommandHandler("alert", set_alert))
    app.add_handler(CommandHandler("myalerts", my_alerts_command))
    app.add_handler(CommandHandler("cancelalert", cancel_alert))
    app.add_handler(CommandHandler("chart", send_price_chart))
    app.add_handler(CommandHandler("admin_stats", admin_stats)) #
Sabon Admin Command

    # Message Handler for unknown commands
    app.add_handler(MessageHandler(filters.COMMAND, unknown))

    # Callback Query Handler for inline keyboard buttons
    app.add_handler(CallbackQueryHandler(button_handler))

    # Job Queue for checking alerts
    app.job_queue.run_repeating(check_alerts, interval=300, first=10)

    logger.info("Bot is starting...")
    app.run_polling(allowed_updates=Update.ALL_TYPES)

if __name__ == "__main__":
    main()
```