```
import os
from telegram import Update
from telegram.ext import (
    Application,
    MessageHandler,
    CommandHandler,
    ContextTypes,
    filters,
)
from mcs import MedicalCoderSwarm
import logging
from dotenv import load_dotenv

load_dotenv()

# Configure logging
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)


class MedicalBot:
    def __init__(self, token):
        # Initialize bot
        self.app = Application.builder().token(token).build()

        # Store conversation context
    self.contexts = {}

    # Setup handlers
    self.app.add_handler(CommandHandler("start", self.start))
    self.app.add_handler(CommandHandler("clear", self.clear))
    self.app.add_handler(
        MessageHandler(
            filters.TEXT & ~filters.COMMAND, self.handle_message
        )
    )

async def start(
    self, update: Update, context: ContextTypes.DEFAULT_TYPE
):
    """Handle the /start command"""
    chat_id = update.effective_chat.id
    self.contexts[chat_id] = []
    await update.message.reply_text(
        "Hello! I'm your medical coding assistant. How can I help you today?"
    )

async def clear(
    self, update: Update, context: ContextTypes.DEFAULT_TYPE
):
    """Clear conversation history"""
    chat_id = update.effective_chat.id
    self.contexts[chat_id] = []

    await update.message.reply_text(

        "Conversation history cleared!"

    )


async def handle_message(

    self, update: Update, context: ContextTypes.DEFAULT_TYPE

):

    """Process incoming messages"""

    chat_id = update.effective_chat.id

    message = update.message.text


    # Initialize context if needed

    if chat_id not in self.contexts:

        self.contexts[chat_id] = []


    # Add message to context

    self.contexts[chat_id].append(f"User: {message}")


    try:

        # Create swarm instance with context

        swarm = MedicalCoderSwarm(

            patient_id=str(chat_id),

            max_loops=1,

            patient_documentation="",

        )
```

```
            # Build complete context
            full_context = "\n".join(
                self.contexts[chat_id][-5:]
            )  # Last 5 messages

            # Get response from swarm
            response = swarm.run(task=full_context)

            # Add response to context
            self.contexts[chat_id].append(f"Assistant: {response}")

            # Keep only last 10 messages in context
            if len(self.contexts[chat_id]) > 10:
                self.contexts[chat_id] = self.contexts[chat_id][-10:]

            # Send response
            await update.message.reply_text(response)

        except Exception as e:
            logger.error(f"Error processing message: {e}")
            await update.message.reply_text(
                "I encountered an error processing your message. Please try again."
            )

    def run(self):
        """Start the bot"""

        logger.info("Starting bot...")

        self.app.run_polling()



if __name__ == "__main__":

    # Replace with your bot token

    TOKEN = os.getenv("TELEGRAM_API_KEY")


    # Create and run bot

    bot = MedicalBot(TOKEN)

    bot.run()
```