```
from telegram import Update, BotCommand
from telegram.ext import (
    ApplicationBuilder,
    CommandHandler,
    MessageHandler,
    ContextTypes,
    filters,
)
import random

# Store player states
player_states = {}

class Player:
    def __init__(self):
        self.level = 1
        self.xp = 0
        self.hp = 100
        self.monster = None
        self.inventory = ["Potion"]
        self.guild = None

    def spawn_monster(self):
        self.monster = {
            "name": random.choice(["Goblin", "Orc", "Demon", "Ice Elf", "Iron Golem"]),
            "hp": random.randint(20, 50) + self.level * 5,
            "attack": random.randint(5, 15) + self.level,
        }

# Command handlers
async def start(update: Update, context: ContextTypes.DEFAULT_TYPE):
    user_id = update.message.from_user.id
    player_states[user_id] = Player()
    await update.message.reply_text(
        "🔥 Welcome, Hunter! Type /hunt to find monsters and begin your journey."
    )

async def hunt(update: Update, context: ContextTypes.DEFAULT_TYPE):
    user_id = update.message.from_user.id
    player = player_states.get(user_id)
    if not player:
        await update.message.reply_text("Type /start to begin your journey!")
        return
```

```
    if player.monster:
        await update.message.reply_text(
            f"You're already fighting a {player.monster['name']} with {player.monster['hp']} HP.\n"
            "Type /attack to fight!"
        )
    else:
        player.spawn_monster()
        await update.message.reply_text(
            f"A wild {player.monster['name']} appeared with {player.monster['hp']} HP!\n"
            "Type /attack to fight!"
        )

async def attack(update: Update, context: ContextTypes.DEFAULT_TYPE):
    user_id = update.message.from_user.id
    player = player_states.get(user_id)
    if not player or not player.monster:
        await update.message.reply_text("No monster to fight. Type /hunt to find one!")
        return

    damage = random.randint(10, 20) + player.level
    player.monster["hp"] -= damage

    if player.monster["hp"] <= 0:
        xp_gained = random.randint(20, 40)
        player.xp += xp_gained
        response = (
            f"You defeated the {player.monster['name']} and gained {xp_gained} XP!\n"
        )
        if player.xp >= player.level * 50:
            player.xp = 0
            player.level += 1
            response += f"🎉 You leveled up! You're now level {player.level}!\n"
        response += "Type /hunt to find another monster."
        player.monster = None
    else:
        monster_attack = random.randint(5, player.monster["attack"])
        player.hp -= monster_attack
        response = (
            f"You dealt {damage} damage. The monster has {player.monster['hp']} HP left.\n"
            f"The {player.monster['name']} attacked you for {monster_attack} damage.\n"
            f"Your HP: {player.hp}\n"
        )
        if player.hp <= 0:
            response += "💀 You have been defeated! Type /start to try again."
```

```
            player_states.pop(user_id)

    await update.message.reply_text(response)

async def stats(update: Update, context: ContextTypes.DEFAULT_TYPE):
    user_id = update.message.from_user.id
    player = player_states.get(user_id)
    if not player:
        await update.message.reply_text("Type /start to begin your journey!")
        return
    await update.message.reply_text(
        f"👨‍⚕️ Level: {player.level}\n"
        f"❤️ HP: {player.hp}\n"
        f"✨ XP: {player.xp}/{player.level * 50}\n"
        f"🎒 Inventory: {', '.join(player.inventory)}"
    )

async def heal(update: Update, context: ContextTypes.DEFAULT_TYPE):
    user_id = update.message.from_user.id
    player = player_states.get(user_id)
    if not player:
        await update.message.reply_text("Type /start to begin your journey!")
        return

    if "Potion" in player.inventory:
        player.hp += 30
        if player.hp > 100:
            player.hp = 100
        player.inventory.remove("Potion")
        await update.message.reply_text(f"🩹 You used a potion. HP: {player.hp}")
    else:
        await update.message.reply_text("You have no potions. Visit the /shop!")

async def inventory(update: Update, context: ContextTypes.DEFAULT_TYPE):
    user_id = update.message.from_user.id
    player = player_states.get(user_id)
    if not player:
        await update.message.reply_text("Type /start to begin your journey!")
        return
    await update.message.reply_text(f"🎒 Your inventory: {', '.join(player.inventory)}")

async def shop(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("🛒 Shop: Potions (50 gold). Type /heal to use potions.")
```

```
async def quests(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("🗺️ You have no active quests yet. Stay tuned!")

async def reset(update: Update, context: ContextTypes.DEFAULT_TYPE):
    user_id = update.message.from_user.id
    player_states.pop(user_id, None)
    await update.message.reply_text("🔄 Your progress has been reset. Type /start to begin
again.")

async def help_command(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("📜 Use these commands to play:\n"
                                    "/hunt, /attack, /stats, /heal, /inventory, /shop, /quests, /reset")

# Additional placeholders for future features
async def leaderboard(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("🏆 Leaderboard coming soon!")

async def equip(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("⚔️ Equip feature coming soon!")

async def unequip(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("⚔️ Unequip feature coming soon!")

async def dungeon(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("🏰 Dungeon feature coming soon!")

async def summon(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("🖤 Summon feature coming soon!")

async def profile(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("🧝 Profile feature coming soon!")

async def skill(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("🌀 Skill system coming soon!")

async def training(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("🏋️ Training coming soon!")

async def guild(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("🏰 Guild system coming soon!")

async def party(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("🧙 Party system coming soon!")
```

```
async def pvp(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("⚔️ PvP duels coming soon!")

async def events(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("🎉 Events coming soon!")

async def set_commands(app):
    commands = [
        BotCommand("start", "Begin your Solo Leveling journey"),
        BotCommand("hunt", "Search for monsters"),
        BotCommand("attack", "Attack the current monster"),
        BotCommand("stats", "View your hunter's stats"),
        BotCommand("heal", "Use potions to recover HP"),
        BotCommand("inventory", "View your items"),
        BotCommand("shop", "Buy items to help in your journey"),
        BotCommand("quests", "View quests and progress"),
        BotCommand("reset", "Reset your stats and progress"),
        BotCommand("help", "Show help and available commands"),
        BotCommand("leaderboard", "Show the top hunters in the game"),
        BotCommand("equip", "Equip a weapon or armor piece"),
        BotCommand("unequip", "Remove an equipped item"),
        BotCommand("dungeon", "Enter a dungeon for bigger rewards"),
        BotCommand("summon", "Summon a shadow soldier to aid you"),
        BotCommand("profile", "View a detailed profile of your hunter"),
        BotCommand("skill", "Use a special skill"),
        BotCommand("training", "Train to increase your stats temporarily"),
        BotCommand("guild", "Join a guild or create one"),
        BotCommand("party", "Form a party with other hunters"),
        BotCommand("pvp", "Challenge another hunter to a duel"),
        BotCommand("events", "See ongoing events in the game world"),
    ]
    await app.bot.set_my_commands(commands)

async def main():
    BOT_TOKEN = 7887245574:AAGbuvy5PKIcNqEmla42UARxVQVl0KaThrY
    app = ApplicationBuilder().token(BOT_TOKEN).build()

    await set_commands(app)

    app.add_handler(CommandHandler("start", start))
    app.add_handler(CommandHandler("hunt", hunt))
    app.add_handler(CommandHandler("attack", attack))
    app.add_handler(CommandHandler("stats", stats))
    app.add_handler(CommandHandler("heal", heal))
```

```
    app.add_handler(CommandHandler("inventory", inventory))
    app.add_handler(CommandHandler("shop", shop))
    app.add_handler(CommandHandler("quests", quests))
    app.add_handler(CommandHandler("reset", reset))
    app.add_handler(CommandHandler("help", help_command))
    app.add_handler(CommandHandler("leaderboard", leaderboard))
    app.add_handler(CommandHandler("equip", equip))
    app.add_handler(CommandHandler("unequip", unequip))
    app.add_handler(CommandHandler("dungeon", dungeon))
    app.add_handler(CommandHandler("summon", summon))
    app.add_handler(CommandHandler("profile", profile))
    app.add_handler(CommandHandler("skill", skill))
    app.add_handler(CommandHandler("training", training))
    app.add_handler(CommandHandler("guild", guild))
    app.add_handler(CommandHandler("party", party))
    app.add_handler(CommandHandler("pvp", pvp))
    app.add_handler(CommandHandler("events", events))

    print("🖤 Solo Leveling bot is running!")
    await app.run_polling()

if __name__ == "__main__":
    import asyncio
    asyncio.run(main())
```