```
# bot.py
import json
import time
from telegram import Update
from telegram.ext import ApplicationBuilder, CommandHandler, MessageHandler,
ContextTypes, filters

# Вставь сюда токен своего бота
TOKEN = "8221262811:AAEl4tw4F9N0yMlG29Ny5Kw1-DprMj0FlbA"

SCORES_FILE = "scores.json"
GAME_DURATION = 30 * 60  # 30 минут
START_TIME = time.time()
GAME_ACTIVE = True

SLOT_POINTS = {
    43: 300,    # 🍋🍋🍋
    64: 400,    # BAR
    85: 500,    # 🍇🍇🍇
    111: 1000   # 777
}

SLOT_NAMES = {
    43: "🍋🍋🍋",
    64: "BAR",
    85: "🍇🍇🍇",
    111: "777"
}

def is_game_active() -> bool:
    return GAME_ACTIVE and (time.time() - START_TIME) < GAME_DURATION

def time_left() -> int:
    remaining = GAME_DURATION - (time.time() - START_TIME)
    return max(0, int(remaining))

def load_scores():
    try:
        with open(SCORES_FILE, "r", encoding="utf-8") as f:
            return json.load(f)
    except FileNotFoundError:
        return {}

def save_scores(scores):
    with open(SCORES_FILE, "w", encoding="utf-8") as f:
        json.dump(scores, f, ensure_ascii=False, indent=2)

def build_top_text() -> str:
```

```
    scores = load_scores()
    if not scores:
        return "Таблица пуста. Никто не играл 🎰."
    sorted_scores = sorted(scores.values(), key=lambda x: x["points"], reverse=True)
    text = "🏆 Финальный топ игроков:\n"
    for i, entry in enumerate(sorted_scores[:5], start=1):
        text += f"{i}. {entry['name']} — {entry['points']} баллов\n"
    return text

async def end_game(context: ContextTypes.DEFAULT_TYPE):
    global GAME_ACTIVE
    GAME_ACTIVE = False
    chat_id = context.job.chat_id
    await context.bot.send_message(chat_id, "⏰ Время игры истекло!\n" + build_top_text())

async def handle_dice(update: Update, context: ContextTypes.DEFAULT_TYPE):
    if not is_game_active():
        await update.message.reply_text("⏰ Игра уже закончилась.")
        return
    if update.message and update.message.dice and update.message.dice.emoji == "🎰":
        value = update.message.dice.value
        user = update.message.from_user
        if value in SLOT_POINTS:
            points = SLOT_POINTS[value]
            scores = load_scores()
            user_id = str(user.id)
            if user_id not in scores:
                scores[user_id] = {"name": user.full_name, "points": 0}
            scores[user_id]["points"] += points
            save_scores(scores)
            combo_name = SLOT_NAMES.get(value, str(value))
            await update.message.reply_text(
                f"🎉 {user.full_name} словил комбинацию **{combo_name}** и получает {points}
баллов!\n"
                f"Текущий счёт: {scores[user_id]['points']}"
            )

async def score(update: Update, context: ContextTypes.DEFAULT_TYPE):
    if not is_game_active():
        await update.message.reply_text("⏰ Игра уже закончилась.")
        return
    user = update.message.from_user
    scores = load_scores()
    user_id = str(user.id)
    if user_id in scores:
        await update.message.reply_text(f"🏅 {user.full_name}, у тебя
{scores[user_id]['points']} баллов.")
    else:
```

```
        await update.message.reply_text("У тебя пока нет баллов. Играй в 🎰 чтобы заработать!")

async def top(update: Update, context: ContextTypes.DEFAULT_TYPE):
    text = build_top_text()
    if not is_game_active():
        text += "\n⏰ Игра завершена!"
    await update.message.reply_text(text)

async def time_cmd(update: Update, context: ContextTypes.DEFAULT_TYPE):
    remaining = time_left()
    if remaining <= 0:
        await update.message.reply_text("⏰ Время игры закончилось!")
    else:
        minutes = remaining // 60
        seconds = remaining % 60
        await update.message.reply_text(f"⌛ До конца игры осталось {minutes} мин {seconds} сек.")

def main():
    app = ApplicationBuilder().token(TOKEN).build()
    app.job_queue.run_once(end_game, when=GAME_DURATION, chat_id=None)
    app.add_handler(MessageHandler(filters.Dice.ALL, handle_dice))
    app.add_handler(CommandHandler("score", score))
    app.add_handler(CommandHandler("top", top))
    app.add_handler(CommandHandler("time", time_cmd))
    app.run_polling()

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