Binance Futures Testnet Trading Bot - Source Code

requirements.txt

```
python-binance==1.0.19
python-dotenv==1.0.1
```

bot/client.py

```
import os
from binance.client import Client
from dotenv import load_dotenv

load_dotenv()

class BinanceFuturesClient:
    def __init__(self):
        api_key = os.getenv("BINANCE_API_KEY")
        api_secret = os.getenv("BINANCE_API_SECRET")

        if not api_key or not api_secret:
            raise ValueError("API keys not found. Please set them in the .env file.")

        self.client = Client(api_key, api_secret, testnet=True)
```

bot/logging_config.py

```
import logging
import os

LOG_DIR = "logs"
os.makedirs(LOG_DIR, exist_ok=True)

def setup_logger():
    logger = logging.getLogger("trading_bot")
    logger.setLevel(logging.INFO)

    if not logger.handlers:
        file_handler = logging.FileHandler(f"{LOG_DIR}/bot.log")
        formatter = logging.Formatter(
            "%(asctime)s - %(levelname)s - %(name)s - %(message)s"
        )
        file_handler.setFormatter(formatter)
        logger.addHandler(file_handler)

    return logger
```

bot/validators.py

```
def validate_symbol(symbol: str):
    if not symbol or len(symbol) < 6:
        raise ValueError("Invalid symbol format (e.g., BTCUSDT)")
    return symbol.upper()

def validate_side(side: str):
    side = side.upper()
    if side not in ["BUY", "SELL"]:
        raise ValueError("Side must be BUY or SELL")
    return side
def validate_order_type(order_type: str):
```

```
    order_type = order_type.upper()
    if order_type not in ["MARKET", "LIMIT"]:
        raise ValueError("Order type must be MARKET or LIMIT")
    return order_type

def validate_quantity(qty: float):
    if qty <= 0:
        raise ValueError("Quantity must be greater than 0")
    return qty

def validate_price(price, order_type):
    if order_type == "LIMIT":
        if price is None or price <= 0:
            raise ValueError("Price must be provided and > 0 for LIMIT orders")
    return price
```

bot/orders.py

```
import logging
from binance.exceptions import BinanceAPIException

logger = logging.getLogger("trading_bot")

class OrderManager:
    def __init__(self, client):
        self.client = client.client

    def place_order(self, symbol, side, order_type, quantity, price=None):
        try:
            logger.info(f"Order Request → {symbol} | {side} | {order_type} | Qty: {quantity} | Price

            if order_type == "MARKET":
                order = self.client.futures_create_order(
                    symbol=symbol,
                    side=side,
                    type="MARKET",
                    quantity=quantity
                )
            else:
                order = self.client.futures_create_order(
                    symbol=symbol,
                    side=side,
                    type="LIMIT",
                    quantity=quantity,
                    price=price,
                    timeInForce="GTC"
                )

            logger.info(f"Order Response → {order}")
            return order

        except BinanceAPIException as e:
            logger.error(f"Binance API Error: {e.message}")
            raise

        except Exception as e:
            logger.error(f"Unexpected Error: {str(e)}")
            raise
```

cli.py

```
import argparse
from bot.client import BinanceFuturesClient
from bot.orders import OrderManager
from bot.validators import *
from bot.logging_config import setup_logger

logger = setup_logger()

def main():
    parser = argparse.ArgumentParser(description="Binance Futures Testnet Trading Bot")
```

```
    parser.add_argument("--symbol", required=True, help="Trading pair symbol, e.g. BTCUSDT")
    parser.add_argument("--side", required=True, help="BUY or SELL")
    parser.add_argument("--type", required=True, help="MARKET or LIMIT")
    parser.add_argument("--quantity", type=float, required=True, help="Order quantity")
    parser.add_argument("--price", type=float, help="Price (required for LIMIT)")

    args = parser.parse_args()

    try:
        symbol = validate_symbol(args.symbol)
        side = validate_side(args.side)
        order_type = validate_order_type(args.type)
        quantity = validate_quantity(args.quantity)
        price = validate_price(args.price, order_type)

        print("\n■ ORDER REQUEST SUMMARY")
        print(f"Symbol   : {symbol}")
        print(f"Side     : {side}")
        print(f"Type     : {order_type}")
        print(f"Quantity : {quantity}")
        if price:
            print(f"Price    : {price}")

        client = BinanceFuturesClient()
        manager = OrderManager(client)

        order = manager.place_order(symbol, side, order_type, quantity, price)

        print("\n■ ORDER PLACED SUCCESSFULLY")
        print(f"Order ID      : {order.get('orderId')}")
        print(f"Status        : {order.get('status')}")
        print(f"Executed Qty  : {order.get('executedQty')}")
        print(f"Average Price : {order.get('avgPrice')}")

    except Exception as e:
        print(f"\n■ ORDER FAILED: {str(e)}")
        logger.error(f"CLI Error: {str(e)}")

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