Trading Bot System File Structure

```
trading_bot_system/
├── main.py                          # Application entry point
├── requirements.txt                 # Python dependencies
├── config/
│   ├── __init__.py
│   ├── settings.py                  # App configuration
│   ├── exchanges.yaml               # Exchange-specific configs
│   └── logging.yaml                 # Logging configuration
│
├── bot_manager/
│   ├── __init__.py
│   ├── orchestrator.py              # Main bot orchestration
│   ├── bot_factory.py               # Creates bot instances
│   ├── lifecycle.py                 # Start/stop/pause bot logic
│   └── strategies/
│       ├── __init__.py
│       ├── base_strategy.py         # Base strategy interface
│       ├── market_making.py         # Example strategy
│       └── arbitrage.py             # Example strategy
│
├── exchanges/
│   ├── __init__.py
│   ├── base_connector.py            # Abstract exchange interface
│   ├── rate_limiter.py              # Distributed rate limiting
│   ├── connectors/
│   │   ├── __init__.py
│   │   ├── binance.py               # Binance spot + futures
│   │   ├── hyperliquid.py           # Hyperliquid futures
│   │   ├── bybit.py                 # Bybit spot + futures
│   │   ├── mexc.py                  # MEXC spot
│   │   ├── gateio.py                # Gate.io spot
│   │   └── bitget.py                # Bitget spot
│   └── websocket/
│       ├── __init__.py
│       ├── ws_manager.py            # WebSocket connection manager
│       └── reconnection.py         # Reconnection logic
│
├── market_data/
│   ├── __init__.py
│   ├── orderbook_manager.py         # Manages all orderbooks
│   ├── orderbook.py                 # Individual orderbook class
│   ├── aggregator.py                # Combined orderbook logic
│   ├── publisher.py                 # Redis pub-sub publishing
```

```
│   └── data_structures.py          # Price level, book structures
│
├── order_management/
│   ├── __init__.py
│   ├── order_manager.py            # Central order coordination
│   ├── order.py                    # Order data class
│   ├── execution_engine.py         # Order placement logic
│   ├── tracking.py                 # Live order tracking (Redis)
│   └── reconciliation.py           # Order state reconciliation
│
├── database/
│   ├── __init__.py
│   ├── connection.py               # PostgreSQL connection
│   ├── redis_client.py             # Redis connection
│   ├── models.py                   # SQLAlchemy models
│   ├── migrations/
│   │   ├── __init__.py
│   │   └── 001_initial_schema.sql
│   └── repositories/
│       ├── __init__.py
│       ├── bot_repository.py       # Bot CRUD operations
│       ├── trade_repository.py     # Trade history
│       └── order_repository.py     # Order history
│
├── api/
│   ├── __init__.py
│   ├── app.py                      # FastAPI application
│   ├── websocket_handler.py        # WebSocket for frontend
│   ├── routes/
│   │   ├── __init__.py
│   │   ├── bots.py                 # Bot management endpoints
│   │   ├── orders.py               # Order endpoints
│   │   ├── market_data.py          # Market data endpoints
│   │   └── health.py               # Health checks
│   └── middleware/
│       ├── __init__.py
│       ├── auth.py                 # Authentication
│       └── cors.py                 # CORS handling
│
├── frontend/
│   ├── package.json
│   ├── src/
│   │   ├── App.js                  # Main React app
│   │   ├── components/
```

```
│   │   │   ├── BotDashboard.js      # Bot status overview
│   │   │   ├── BotControls.js       # Start/stop controls
│   │   │   ├── OrderBook.js         # Orderbook visualization
│   │   │   ├── TradeHistory.js      # Trade history table
│   │   │   └── ConfigEditor.js      # Bot configuration
│   │   ├── services/
│   │   │   ├── api.js               # Backend API calls
│   │   │   └── websocket.js         # WebSocket connection
│   │   └── utils/
│   │       └── formatters.js        # Price/number formatting
│   └── public/
│       └── index.html
│
├── utils/
│   ├── __init__.py
│   ├── logger.py                    # Logging utilities
│   ├── exceptions.py                # Custom exceptions
│   ├── validators.py                # Data validation
│   ├── formatters.py                # Price/decimal formatting
│   └── metrics.py                   # Performance metrics
│
├── tests/
│   ├── __init__.py
│   ├── conftest.py                  # Pytest configuration
│   ├── unit/
│   │   ├── test_orderbook.py
│   │   ├── test_rate_limiter.py
│   │   └── test_strategies.py
│   ├── integration/
│   │   ├── test_exchange_connectors.py
│   │   └── test_order_flow.py
│   └── fixtures/
│       ├── sample_orderbook.json
│       └── sample_trades.json
│
├── scripts/
│   ├── setup_database.py           # Initialize PostgreSQL
│   ├── setup_redis.py              # Initialize Redis
│   ├── backfill_data.py            # Historical data import
│   └── health_check.py             # System health monitoring
│
├── docker/
│   ├── Dockerfile                  # Main application
│   ├── docker-compose.yml          # Full stack setup
```

```
│   ├── postgres/
│   │   └── init.sql                # Database initialization
│   └── redis/
│       └── redis.conf              # Redis configuration
│
└── docs/
    ├── README.md                   # Project overview
    ├── SETUP.md                    # Installation guide
    ├── API.md                      # API documentation
    └── ARCHITECTURE.md             # Architecture overview
```

Key File Responsibilities

main.py: Application startup, orchestrates all services **bot_manager/orchestrator.py**: Central bot control, manages bot lifecycle **exchanges/base_connector.py**: Unified exchange interface all connectors inherit **market_data/orderbook_manager.py**: Maintains all orderbooks, publishes to Redis **order_management/order_manager.py**: Central order coordination across exchanges **api/app.py**: FastAPI server for frontend communication **database/models.py**: SQLAlchemy models for PostgreSQL tables

Configuration Structure

config/settings.py:

```python
EXCHANGES = {
    'binance': {'api_key': '...', 'secret': '...'},
    'bybit': {'api_key': '...', 'secret': '...'}
}

REDIS_CONFIG = {'host': 'localhost', 'port': 6379}
POSTGRES_CONFIG = {'host': 'localhost', 'port': 5432}

BOT_CONFIGS = {
    'market_maker_1': {
        'strategy': 'market_making',
        'exchanges': ['binance', 'bybit'],
        'symbols': ['BTCUSDT', 'ETHUSDT']
    }
}
```

This structure separates concerns clearly while maintaining Python-first approach with asyncio for performance.