Software Design Specification

for

Bybit Cryptocurrency Derivatives Trading Bot

Version 1.1 approved

Prepared by Steven Wilkins

Eastern Oregon University – CS401 Capstone

August 4, 2022

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Steven Wilkins	8-4-2022	Initial document creation	1.0
Steven Wilkins	1-5-2023	Fixed “Error: Bookmark not defined” in table of contents. Added example image for section 3.6 .	1.1

1. Introduction

This is the design document for the Bybit Cryptocurrency Derivatives Trading Bot (BCDTB). The purpose of the BCDTB is to automate trading by using a predetermined algorithmic strategy. For more information, the BCDTB is outlined in more detail inside the current project's requirements document.

2. Design Specification

2.1 Programming Language

- Python 3.8 with Pip
- Pip packages: ccxt, pandas, numpy, mplfinance, schedule, warnings, time, datetime, pprint, os, glob

2.2 Development Environment

2.2.1 Anaconda Navigator 3

Anaconda Navigator makes setting up a python development environment extremely easy and makes multiple development tools available to connect to the same environment. The following tools will be used inside the Anaconda environment.

- **Jupyter Notebook** – This is an indispensable troubleshooting tool for Python code.
- **Spyder IDE** – Python IDE available inside Anaconda with a myriad of helpful tools available.

3. BCDTB

3.1 File:README.md

User instructions are documented within the README.md file.

3.2 File:requirements.txt

Contains a list of pip package requirements for the python environment. Can be installed by entering the BCDTB's root directory, activating a dedicated python3 environment and running the command:

Python3 -m pip install -r requirements.txt

3.3 Script:bot.py

Imported Modules:	config.py	ccxt	myasset.py
	Warnings	time	schedule

This is the main python file that gets called to run the BCDTB, which opens a connection to the user's Bybit account and initializes all the required global variables with user supplied values from '/settings/config.py'.

Global Variables

Variable Name	Parameters Assigned From '/settings/config.py'
exchange	config.API_KEY; config.SECRET_KEY; config.SANDBOX
symbol	config.SYMBOL
timeframe	config.TIME_FRAME
amount	config.AMOUNT
leverage	config.LEVERAGE
stoploss	config.STOP_LOSS
takeprofit	config.TAKE_PROFIT

Functions

Function Name	Description
set_exchange	Initializes a connection to the user's Bybit account with the config.API_KEY, config.SECRET_KEY, and config.SANDBOX. If config.SANDBOX is set to True, then exchange will be connected to the Bybit testnet. Returns the exchange instance.
prepare_assets	Parameters: exchange, symbol, timeframe, amount, leverage, stoploss, and takeprofit Action: Creates one or more Myasset objects using the variables initialized from config.py and stores them in a list.
set_leverage	Initializes leverage with config.LEVERAGE
run_bot	Calls Myasset.run_bot for each created Myasset object.

3.4 Module:BCDTB/python/myasset.py

Imported Modules:	pandas	ccxt	mplfinance
	Warnings	time	datetime
	Numpy	os	pprint
	glob		

This module contains only one class, called Myasset. The Myasset constructor initializes an object for the symbol passed in as a parameter along with all the attributes and settings that go along with it. At this time, the trading strategy is coded into the Mysset class, but will be moved to a separate module in the future.

Global Class Variables

Variable Name	Parameters Initialized From Myasset Constructor
self.exchange	Initialized with the exchange parameter passed to the constructor.
self.symbol	Initialized with the symbol parameter passed to the constructor.
self.timeframe	Initialized with the timeframe parameter passed to the constructor.
self.amount	Initialized with the amount parameter passed to the constructor.
self.leverage	Initialized with the leverage parameter passed to the constructor.
self.stoploss	Initialized with the stoploss parameter passed to the constructor.
self.takeprofit	Initialized with the takeprofit parameter passed to the constructor.
self.df	Initialized as an empty pandas dataframe.
self.trade_amount	Initialized with the amount parameter passed to the constructor.

Functions

Function Name	Description
__repr__	Returns string representation of Myasset. (self.exchange, self.symbol, self.amount, self.leverage, self.timeframe, self.stoploss, self.takeprofit)
get_df	Returns self.df.
__get_last_ticker	IReturns current ticker for received symbol.
__get_wallet_balance	Returns total and available wallet balance for received symbol in USD.
__set_trade_amount	Sets trade amount to either percentage of available balance or specific dollar amount, depending on type of self.amount variable.
__create_ohlcv_df	Creates pandas dataframe with the bars fetched from the exchange.
__plot_chart	Plots a series of charts with indicators and saves them inside a generated directory called charts.
__tr	Calculates true range.
__atr	Calculates average true range.
__supertrend	Calculates supertrend using average true range calculated with the period and atr_multiplier passed in as a parameter.

3.5 File:BCDTB/settings/config.py

Settings must be configured in a file named config.py. Examples of how to do this are presented below.

Set API Key and Secret

```
API_KEY = 'your-api-key'
SECRET_KEY = 'your-api-secret'
```

Set Sandbox (Testnet)

If the API key is for the Bybit testnet, set SANDBOX to True. If using the Live exchange, set SANDBOX to False.

```
SANDBOX = True
```

Set Symbol

Go to https://www.bybit.com/data/markets/ for full list of available market symbols.

If the Trader only wants to trade one market, they should enter the SYMBOL as a string in quotes.

```
SYMBOL = 'BTC/USD'
```

If trading multiple SYMBOLs is desired, provide a list of strings.

```
SYMBOL = ['BTC/USD','ETH/USD']
```

If the Trader's funds are in USDT, and they want to scan every cryptocurrency that pairs with USDT, just provide 'USDT' as a string.

```
SYMBOL = 'USDT'
```

Set Timeframe

Bybit makes OHLCV data of multiple timeframes available for setting TIME_FRAME with. A list of these and the string by which to reference them can be found below.

Time Frame	Reference String
1 Minute	'1m'
3 Minute	'3m'
5 Minute	'5m'
15 Minute	'15m'
30 Minute	'30m'
45 Minute	'45m'
1 Hour	'1h'
2 Hour	'2h'
4 Hour	'4h'

6 Hour	'6h'
12 Hour	'12h'
1 Day	'1d'
1 Week	'1w'
1 Month	'1M'

Below is an example of setting TIME_FRAME to 1 hour in config.py.

```
TIME_FRAME = '1h'
```

Set Position Amount

If it is desired to enter into every position with a specific dollar amount, then assign that value as an integer.

```
AMOUNT = 100
```

If it is desired to use a percentage of the available account, then assign the percentage as a string value in quotes.

```
AMOUNT = '30'
```

Set Leverage

Set LEVERAGE with an integer. For 5x leverage, set LEVERAGE to 5.

```
LEVERAGE = 5
```

Set Stoploss

STOPLOSS is set with percent as a decimal. If the Trader wants to restrict the risk on the position to 15 percent, they would assign .15, i.e., 15/100.

```
STOPLOSS = .15
```

Take Profit

TAKEPROFIT is set with percent as a decimal. If the Trader wants to take profit on the position when it reaches 35 percent, they would assign .35, i.e., 35/100.

```
TAKEPROFIT = .35
```

3.6 GenDir:BCDTB/charts/

The BCDTB generates a chart image for each symbol being traded. It doesn't over-write the previous image. It numbers each new image and groups them in a directory unique to each symbol. Every symbol directory is grouped together under one parent directory (BCDTB/charts/). An example of the full path for a BTC/USD chart would be BCDTB/charts/BTCUSD-charts/BTCUSD-chart(1).png. This is designed to prevent accidental loss of data, but if left unmanaged over an extended period of time it will certainly take up an unacceptable amount of storage on the user's drive. It is recommended to either delete these images when they are no longer needed or transfer them to another storage device.

3.6.1 GenFile:BCDTB/charts/BTCUSD-charts/BTCUSD-chart(8).png