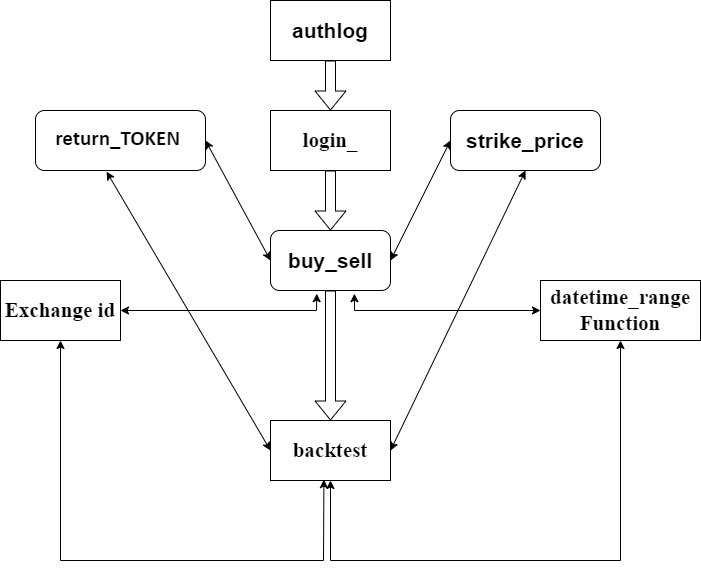
Packages Used:

1. pandas
2. numpy
3. py5paisa
4. mplfinance
5. matplotlib
6. datetime

Resources

1. <https://github.com/matplotlib/mplfinance>
2. <https://github.com/5paisa/py5paisa>

Flow Chart: -



User’s credentials:

User can get this in the 5paisa trading website. And Datatype of all credentials are in string

this includes {App name, app source, user id, password, user key, Encryption key: This will be

in Dictionary} and {email id Password and DOB: this will be declared in function for login

purpose through API’s}

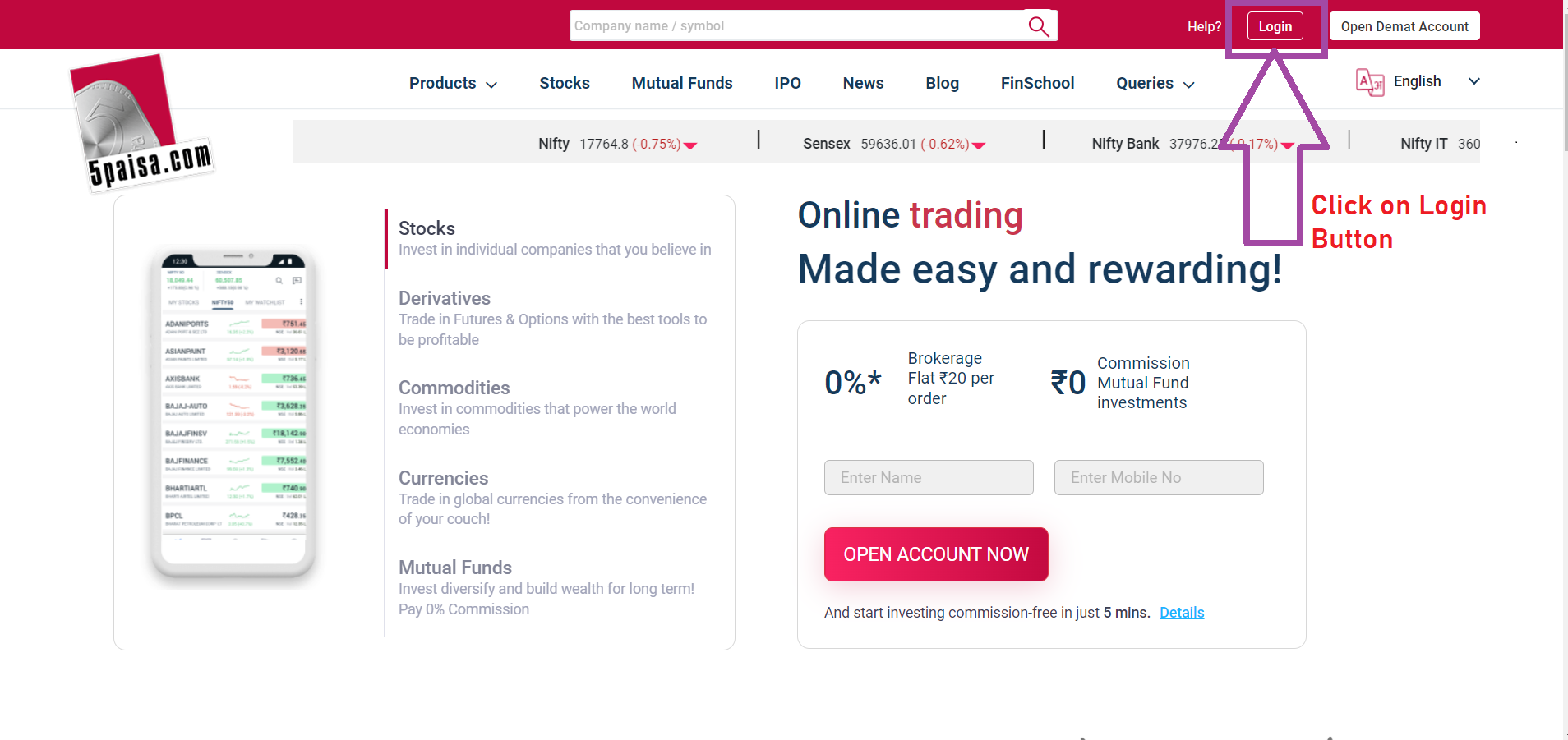
"APP\_NAME": "5P56198212",  
 "APP\_SOURCE": "7312",  
 "USER\_ID": "bedaMQlrqg",  
 "PASSWORD": "IBMfbkUlj65",  
 "USER\_KEY": "maNpe12f7OS212y31zGI8ToIV5zMgBWmousNA",  
 "ENCRYPTION\_KEY": "UEjdasrsadd2DVsdOda54tH5SYwp"

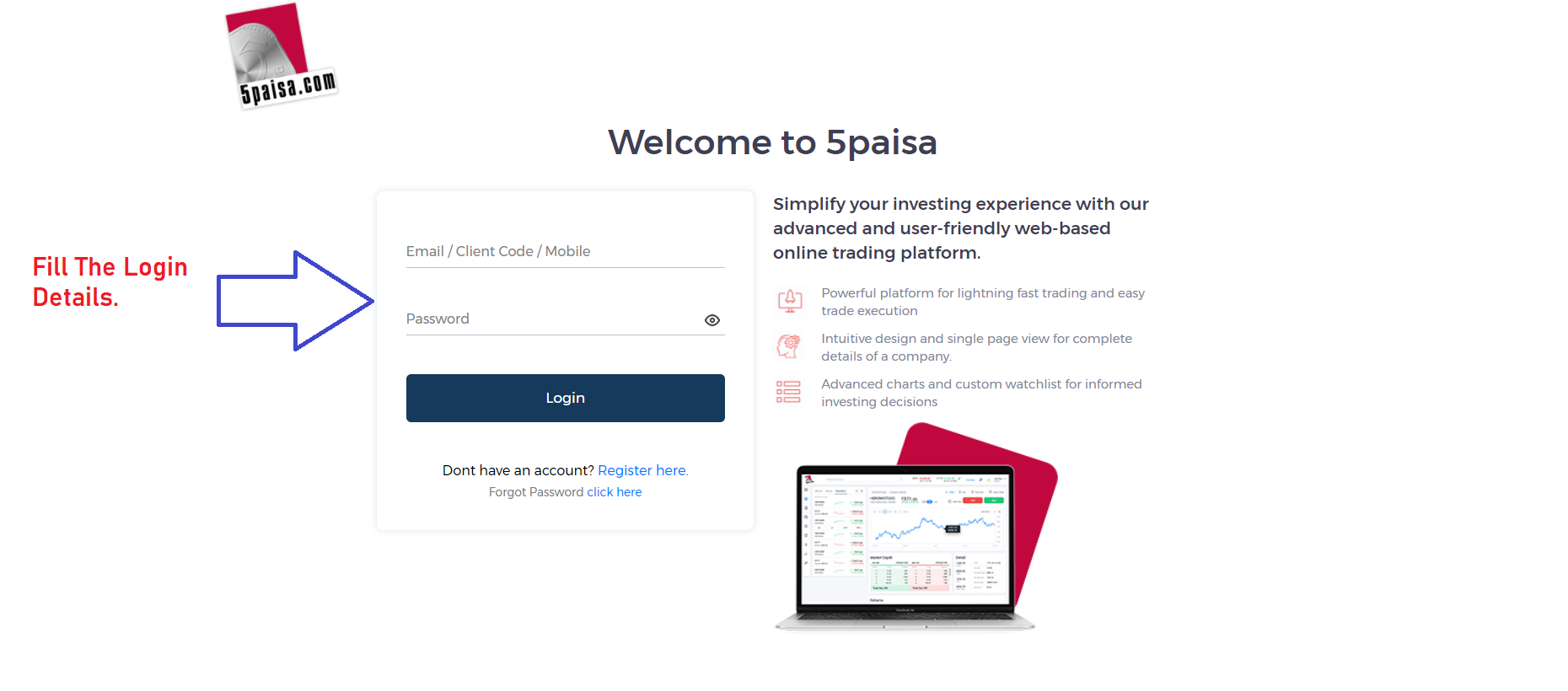
email="xyz@gmail.com", password of the user’s 5paisa account ="xyz”

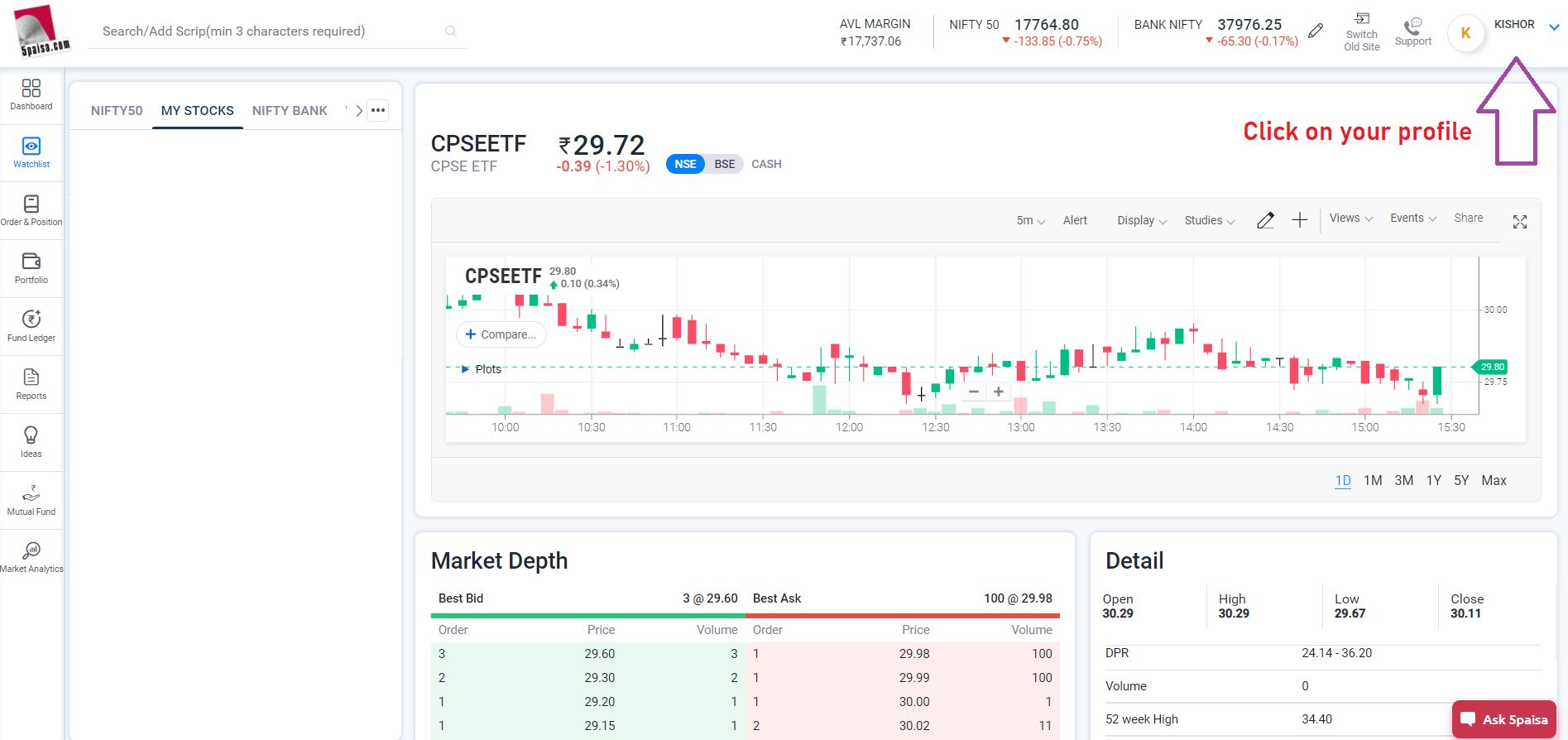
date of birth {year/month/day} = "1234567"

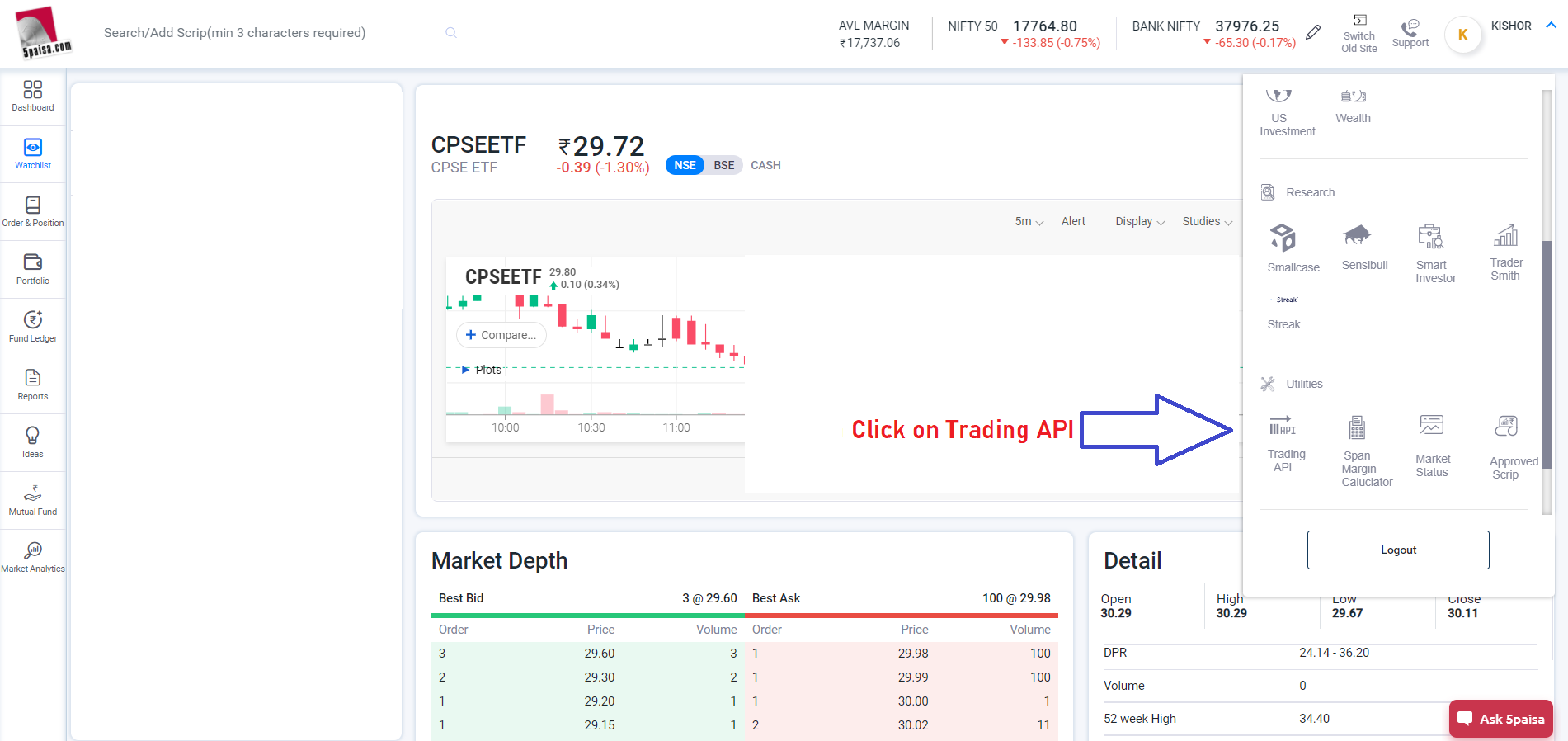
So, for example this will look like this in function

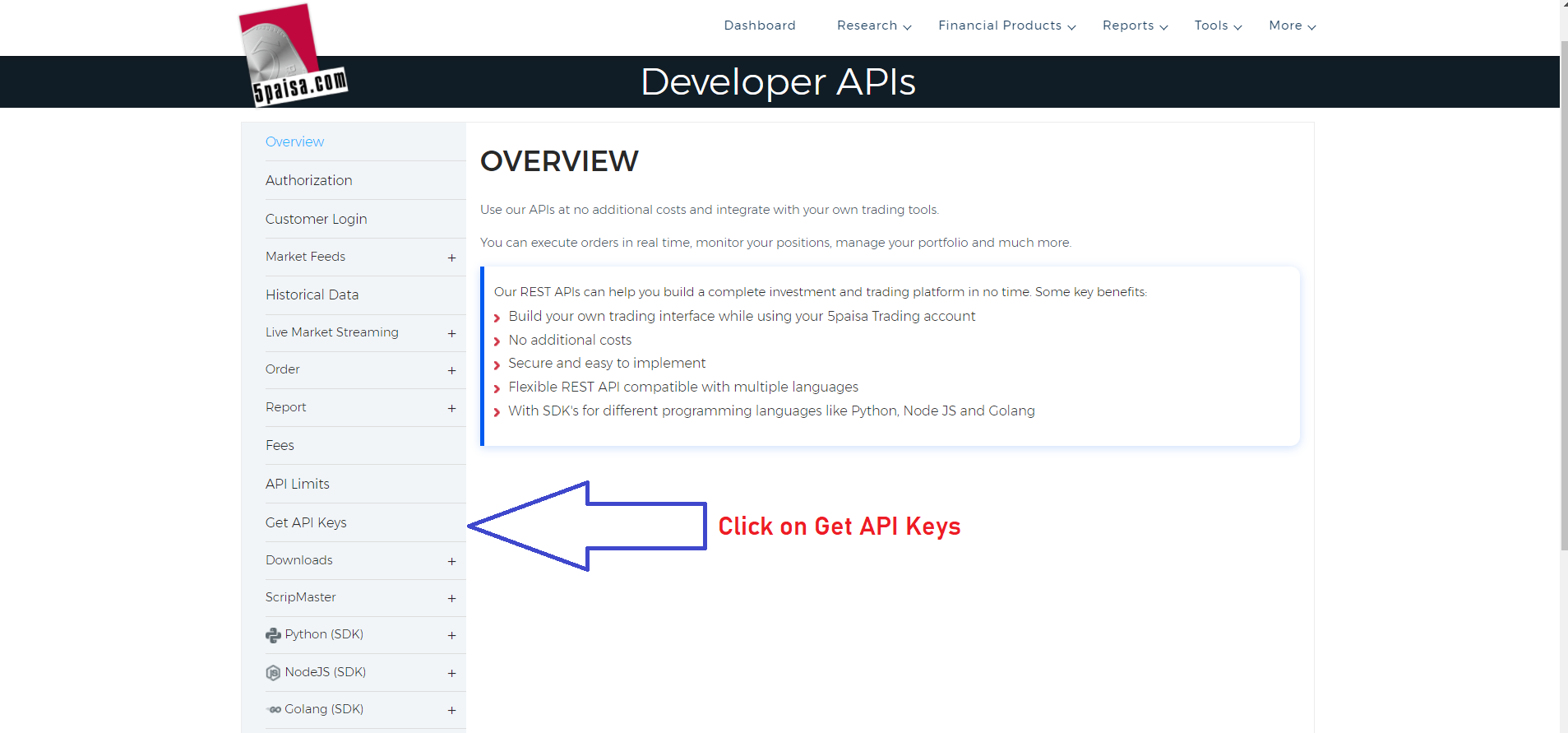
def auth\_():  
 cred = {  
 "APP\_NAME": "5P56798712",  
 "APP\_SOURCE": "7311",  
 "USER\_ID": "beTMrRQlrqg",  
 "PASSWORD": "IBMfbkUlj65",  
 "USER\_KEY": "maNpelRf7OS2yzGI8ToIV5zMgBWmouNA",  
 "ENCRYPTION\_KEY": "UEjzxjrpwi2DVWfOa3GM4L454tH5SYwp"  
 }  
   
 client = FivePaisaClient(email="abcd@gmail.com", passwd="xxxxxxx",  
 dob="11244119", cred=cred)  
  
 client.login()

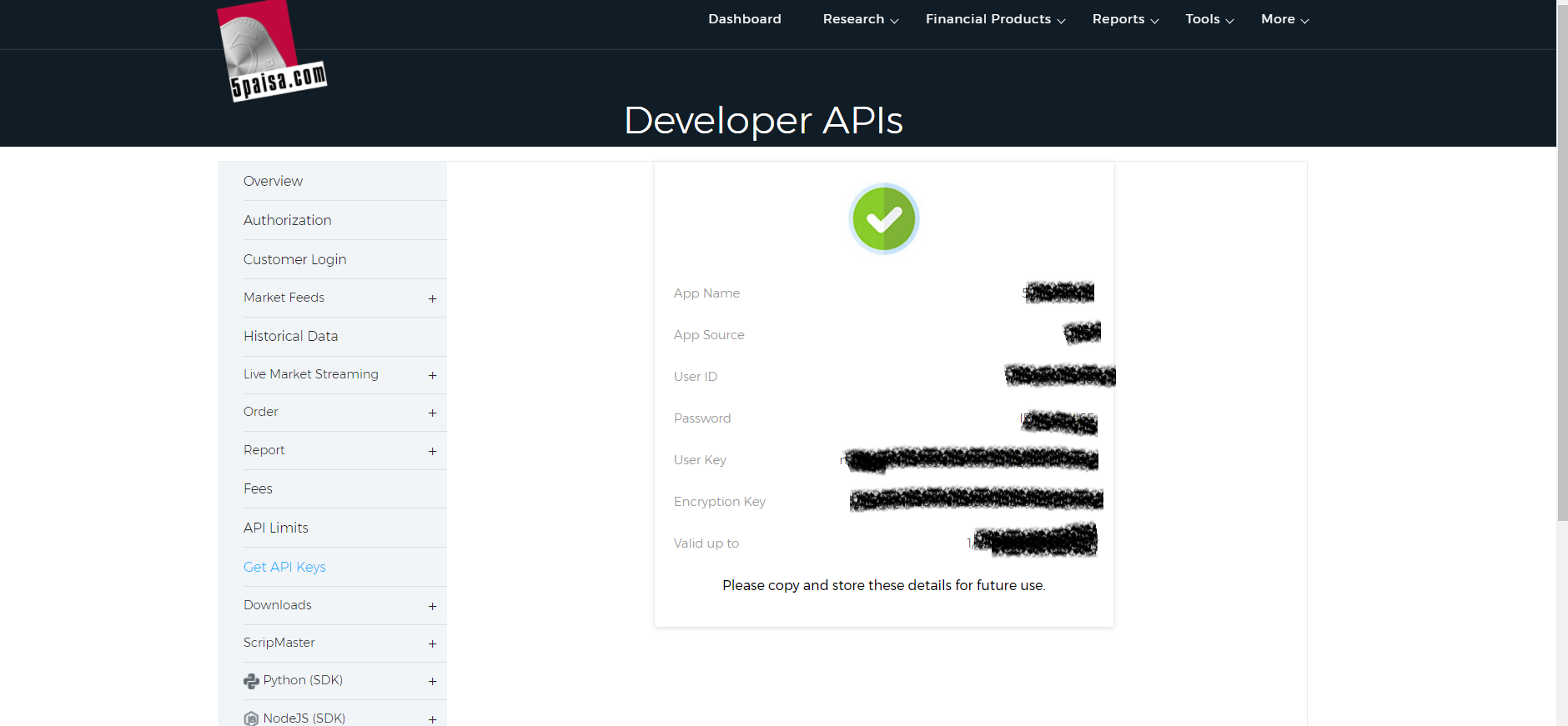
**How to get these login credentials aka API?**

**Fill the Login details**

**Click On your profile and scroll down**

**After Scroll down you will see trading API click on it.**

**Click on get API keys**

**And Here you will find all the credential’s, copy and paste it at same location.**

In this Algo we take, at the money strike Price means if we get a buy single at particular value for e.g., 39650, let’s say this is buying value of option so in order to get strike price at the money we use following user defined function which will return 39600 int value

def strike\_price(snum):  
 base = 100  
 strike\_f = base \* round(snum/base)  
 print(strike\_f)  
 return strike\_f

Exchange id, this is a unique number provided by broker to track the order or trade status so that if we want to cancel the trade or modify the trade (for modifying stoploss especially) we can pass this as a parameter in there modify function i.e., client.modify\_order() , whenever we place an order through API , by this function:- client.place\_order, this returns a dictionary which have exchange id as blank or null value, cause broker don’t provide exchange code immediately after placing a order but they do provide broker code or order id so with the help of order id we can get exchange id , (note: - there is only exchange id parameter to modify the order, we can’t pass order id or broker id to track the order)

So, when we place an order with this client.place\_order() function. It returns this dictionary

{'BrokerOrderID': 576977910, 'ClientCode': '56798712', 'Exch': 'N', 'ExchOrderID': '0', 'ExchType': 'D', 'LocalOrderID': 0, 'Message': 'Order rejected by RMS. Quantity should be in multiple of 25', 'RMSResponseCode': -95, 'ScripInfo': '53963', 'Status': 1, 'Time': '/Date(1637260200000+0530)/'}

1.1

As you can see there is no exchange id, so in order to get it we have user defined function as return\_EXCHANGECODE :-

def return\_EXCHANGECODE(bidict):  
 bidict1 = bidict['BrokerOrderID']  
 jk = client.order\_book() # module function (explanation for this function is below)  
 for i in range(len(jk)):  
 if str(jk[i]['BrokerOrderId']) == str(bidict1):  
 print('exchange id ',jk[i]['ExchOrderID'])  
 return jk[i]['ExchOrderID']

client.order\_book() this is a module function which return list with dictionary elements for e.g.,

this is a list of two trade;

[{'AHProcess': 'N', 'AfterHours': 'N', 'AtMarket': 'N', 'BrokerOrderId': 576975686, 'BrokerOrderTime': '/Date(1637305818340+0530)/', 'BuySell': 'B', 'DelvIntra': 'I', 'DisClosedQty': 1, 'Exch': 'N', 'ExchOrderID': '0', 'ExchOrderTime': '/Date(315513000000+0530)/', 'ExchType': 'D', 'MarketLot': 0, 'OldorderQty': 0, 'OrderRequesterCode': '56798712 ', 'OrderStatus': 'Rejected By 5P', 'OrderValidUpto': '19 Nov 2021', 'OrderValidity': 0, 'PendingQty': 1, 'Qty': 1, 'Rate': 1, 'Reason': 'Order rejected by RMS. Limit price should be within circuit limit(17.8 - 324.6)', 'RemoteOrderID': '1', 'RequestType': 'P', 'SLTriggerRate': 0, 'SLTriggered': 'N', 'SMOProfitRate': 0, 'SMOSLLimitRate': 0, 'SMOSLTriggerRate': 0, 'SMOTrailingSL': 0, 'ScripCode': 48594, 'ScripName': 'NIFTY 20 Jan 2022 PE 16600.00', 'TerminalId': 0, 'TradedQty': 0, 'WithSL': 'N'},

{'AHProcess': 'N', 'AfterHours': 'N', 'AtMarket': 'N', 'BrokerOrderId': 576977910, 'BrokerOrderTime': '/Date(1637308874377+0530)/', 'BuySell': 'B', 'DelvIntra': 'I', 'DisClosedQty': 1, 'Exch': 'N', 'ExchOrderID': '0', 'ExchOrderTime': '/Date(315513000000+0530)/', 'ExchType': 'D', 'MarketLot': 0, 'OldorderQty': 0, 'OrderRequesterCode': '56798712 ', 'OrderStatus': 'Rejected By 5P', 'OrderValidUpto': '19 Nov 2021', 'OrderValidity': 0, 'PendingQty': 1, 'Qty': 1, 'Rate': 1, 'Reason': 'Order rejected by RMS. Quantity should be in multiple of 25', 'RemoteOrderID': '1', 'RequestType': 'P', 'SLTriggerRate': 0, 'SLTriggered': 'N', 'SMOProfitRate': 0, 'SMOSLLimitRate': 0, 'SMOSLTriggerRate': 0, 'SMOTrailingSL': 0, 'ScripCode': 53963, 'ScripName': 'BANKNIFTY 25 Nov 2021 PE 38000.00', 'TerminalId': 0, 'TradedQty': 0, 'WithSL': 'N'}]

So the parameter of return\_EXCHANGECODE is a dictionary, this parameter is nothing but the return of the client.place\_order() (1.1) function so we are just equating a element of dictionary to another element of dictionary for e.g.,

element of client.place\_order() dictionary will be 'BrokerOrderID': 576977910,

{'BrokerOrderID': 576977910, 'ClientCode': '56798712', 'Exch': 'N', 'ExchOrderID': '0', 'ExchType': 'D', 'LocalOrderID': 0, 'Message': 'Order rejected by RMS. Quantity should be in multiple of 25', 'RMSResponseCode': -95, 'ScripInfo': '53963', 'Status': 1, 'Time': '/Date(1637260200000+0530)/'}

And another element will be of client.order\_book() that is 'BrokerOrderId': 576977910,

{'AHProcess': 'N', 'AfterHours': 'N', 'AtMarket': 'N', 'BrokerOrderId': 576977910, 'BrokerOrderTime': '/Date(1637308874377+0530)/', 'BuySell': 'B', 'DelvIntra': 'I', 'DisClosedQty': 1, 'Exch': 'N', 'ExchOrderID': '0', 'ExchOrderTime': '/Date(315513000000+0530)/', 'ExchType': 'D', 'MarketLot': 0, 'OldorderQty': 0, 'OrderRequesterCode': '56798712 ', 'OrderStatus': 'Rejected By 5P', 'OrderValidUpto': '19 Nov 2021', 'OrderValidity': 0, 'PendingQty': 1, 'Qty': 1, 'Rate': 1, 'Reason': 'Order rejected by RMS. Quantity should be in multiple of 25', 'RemoteOrderID': '1', 'RequestType': 'P', 'SLTriggerRate': 0, 'SLTriggered': 'N', 'SMOProfitRate': 0, 'SMOSLLimitRate': 0, 'SMOSLTriggerRate': 0, 'SMOTrailingSL': 0, 'ScripCode': 53963, 'ScripName': 'BANKNIFTY 25 Nov 2021 PE 38000.00', 'TerminalId': 0, 'TradedQty': 0, 'WithSL': 'N'}]

if str(jk[i]['BrokerOrderId']) == str(bidict1): if this condition gets satisfy then it well returns exchange id

As you can now clearly see both BrokerOrderId gets match then return\_EXCHANGECODE function will return exchange id from order book (return jk[i]['ExchOrderID'] )

**return\_TOKEN Function**

This Is a user defined function (return token) which return token code aka script code and this script code is used as a parameter in client.place\_order() function, script code is a code of particular contract in option’s for e.g., Bank Nifty 18th NOV 39000. (note 39000 is a strike price) this whole thing is packed in 4–9 digit of code and that code is script code or scrip code, as you can see return token has a strike price as a parameter (which is strike\_p). and pc is nothing but which contract or option type you want to buy, i.e., Put or call

def return\_TOKEN(pc,strike\_p):  
 from datetime import date  
 client = authlog()  
 today = date.today()  
 offset = (today.weekday() - 3)  
 mdate = today - timedelta(days=offset)  
  
 month={  
 "01":'JAN',  
 "02":'FEB',  
 "03":'MAR',  
 "04":'APR',  
 "05":'MAY',  
 "06":'JUN',  
 "07":'JUL',  
 "08":'AUG',  
 "09":'SEP',  
 "10":'OCT',  
 "11":'NOV',  
 "12":'DEC'  
 }  
  
 date = mdate.day  
 iyear = mdate.year  
 imon = mdate.month  
 mon = month[f'{iyear}{imon}{date}'[4:6]]  
  
 symbol = "BANKNIFTY".upper()  
  
 strike\_f = "{:.2f}".format(float(strike\_price(strike\_p)))  
  
 PE\_CE = pc  
  
 sym=f'{symbol} {25} {mon} {iyear} {PE\_CE} {strike\_f}'  
 print(sym)  
 day\_year\_mon = str(iyear) + str(imon)+str(date)  
 print('dayyearmon:',day\_year\_mon)  
 req=[{"Exch": "N", "ExchType": "D", "Symbol": sym, "Expiry": 20211125, "StrikePrice":strike\_f, "OptionType" : f'{PE\_CE}'}]  
 res = client.fetch\_market\_feed(req)  
 token=res['Data'][0]['Token']  
 return token

**login\_ function**

So login\_ function is a user defined function and it takes day as a argument so that we can get a dataframe from client.historical\_data() (module defined function), client.historical\_data() takes 6 arguments in which ‘N’ stands for NSE, ‘C’ stands for Commodity or cash, ‘999920005’ this is a script code, '5m' means time frame, e.g., 5 minute, 10 minute, 1 minute and so on, here m stands for minute, so last and second last argument means date range form to - where, , f'2021-{mon}-{dayf}', f'2021-{mon}-{dayf}' here mon is a global variable and dayf is a argument, this whole means we are taking one day data of 5 miniute time frame of Nifty bank of NSE {National Stock Exchange}.

def login\_(dayf):  
 df1 = None  
 try:  
 global client  
 client = authlog()  
 df1 = client.historical\_data('N', 'C', 999920005, '5m', f'2021-{mon}-{dayf}', f'2021-{mon}-{dayf}')  
 print(df1)  
 except Exception as e:  
 print(e)  
 return df1

here we return dataframe.

**buy\_sell(date) Funtion**

We are going to calculate an initial buy and a sell signal. So this a user defined function that takes date as a argument to get dataframe from login\_(dayf):, we are inheriting the functions.

def buy\_sell(date)

df = login\_(date)

< SOME CALCULATION >

test\_order = Order(order\_type='S', exchange='N', exchange\_segment='D', scrip\_code=48595,  
 quantity=25, price=0, is\_intraday=True) # explanation for this is given down in bactest . function  
 ipsc = client.place\_order(test\_order)

return df

after calculation and placing order It will return a dataframe.

**def backtest(date) Function**

After getting buy sell signals from buy\_sell(date) function we again going to pass the dataframe to next user defined function which will put a stop loss and further signals. This is the last function of the code

def backtest(date):  
 df = buy\_sell(date)

<SOME CALCULATION >

test\_order = Order(order\_type='S', exchange='N', exchange\_segment='D', scrip\_code=48595,  
 quantity=25, price=0, is\_intraday=True)  
  
 ipsc = client.place\_order(test\_order)

so this is how we are going to place a order, Order() is a class, so order\_type can be buy or sell represented by capital B or S respectively datatype as string, exchange means NSE OR BSE, N for NSE, exchange segment means derivative or equity so ‘D’ stands for derivative. Scrip code is contract code. quantity :Quantity to buy, price=0 means buy or sell at market price, is\_intraday= true, means day trading and not a delivery or long term holding.

**datetime\_range Function**

def datetime\_range(start, end, delta):  
 current = start  
 while current < end:  
 yield current  
 current += delta

datetime\_range Is a user defined function and it is used before placing order and it returns list of 5min interval, after calling a function through this way,

dts = [dt.strftime('%Y-%m-%dT%H:%M:00') for dt in  
 datetime\_range(datetime(yr1, mon, cday, 9), datetime.now(), #cday, mon, yr1 are global   
 timedelta(minutes=5))] # variable’s mon=month,  
 # yr1 = year, cday= current day

we get,

['2021-11-21T09:00:00', '2021-11-21T09:05:00', '2021-11-21T09:10:00', '2021-11-21T09:15:00', '2021-11-21T09:20:00', '2021-11-21T09:25:00', '2021-11-21T09:30:00', '2021-11-21T09:35:00', '2021-11-21T09:40:00', '2021-11-21T09:45:00', '2021-11-21T09:50:00', '2021-11-21T09:55:00', '2021-11-21T10:00:00', '2021-11-21T10:05:00', '2021-11-21T10:10:00', '2021-11-21T10:15:00']

It will only return the element: - '2021-11-21T09:05:00', after 9:05 or at 9:05 so before 9:05 we will have list which will look like this: - ['2021-11-21T09:00:00'].

we are only interested in last second element of the list which we will be equating with dataframe index which is also a datetime index. We used this functions to avoid placing of order in for loop else it will place order each time we run the code. so this function is used in such a way that when we get signal to buy it will first check whether it is at current time or not by picking a element form index (we get index from dataframe ) and equating with a last second element of returned list by function (datetime\_range(start, end, delta)) so by this we are secured from placing random orders.

if df['Datetime'][i] == dts[-2]:  
 test\_order = Order(order\_type='B', exchange='N', exchange\_segment='D', scrip\_code=48594,  
 quantity=25, price=0, is\_intraday=True)  
 icsc = client.place\_order(test\_order)

**Summary**

we first use authlog function to login and by using authlog function we authorize and get dataframe from login\_ function, After that we use buy\_sell function to generate and place order and for further signal we give that dataframe to another function which is backtest function and this function put a stoploss and order according to algorithm.