**Backend guide**

A unit that performs simple operations that can connect with the back-end front-end and signaler.

Since there is currently no buy-sell data coming from the signal, buy-sell transactions can be executed manually from the frontend. Buying and selling is done based on the coin name entered in the buy-sell input field.

Basic functionality for the end-to-end connection between the signaler and the backend has been added. Additionally, for example, the current open, close, and ts values for the eth\_usdt coin are passed to the signaler.

On the front-end side, the market values of the coins and the value of the selected coin can be withdrawn.

The account balance of the bydfi user is displayed with the get\_ballance button.(in start.html screen)

**İmportant!!**

1. The new user must first open the application if they want the program to automatically fill in the mail encoding. If not, they can skip these steps and enter "scan" in the final step of the login process.
2. The steps are as follows:

**1. Go to the Azure App Registration page.**

• https://portal.azure.com

• Sign in with your Microsoft account.

• Go to Azure Active Directory → App registrations.

• Click the name of the application you created earlier (the application you registered for the Graph API).

**2. Go to the Certificates & Secrets section.**

• Click Certificates & Secrets in the left menu.

**3. Create a new client secret.**

• Click the + New client secret button.

• Enter a description (for example: Graph API Python).

• For the expiration period, you can select 1 year or Never.

• Click the Add button.

**4. Copy the generated password.**

• After creating it, you'll see a long password in the Value section (example: hsf8Q~ZpC79db9\_\*\*\*\*).

• Copy this immediately, as you won't see it again once you exit the page.

• In the Python code:

**5 copy keys**

CLIENT\_SECRET = "the\_secret\_you\_copied\_here" (CLIENT\_SECRET ) and paste it into the section.

And copy the Application (client) ID in overview, it is CLIENT\_ID in code.

**6 give permisson**

Then you need to give permisson in API permission->Microsoft graph->app permiison

Mail.read,

offline\_access,

User.Read

**7 auth prev**

When creating an app, enable Authentication (Preview). Azure will prompt you for a URL, so paste it there. **You need to select with Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)!!!**

<https://login.microsoftonline.com/common/oauth2/nativeclient>

**finaly**

Run the listen\_mail.py in Project (you need open comment line 51) When you first run this function you will see a URL in the terminal, copy and paste it in a new tab in the browser where azure is open.

setup

Enter cmd in the project location,

1- pip install -r requirements.txt

2- Run venv\Scripts\activate and if the venv extension appears in cmd, the process was successful.

3- To run the code, simply type Python app.py.

4- The terminal port can be activated after you click the start button and complete the login process in the Chrome browser. (If you receive an error during the login steps or if it takes too long, close and restart app.py.)

5- After the login step is complete, you can perform any operation you want on the 5000 port interface. However, be aware that page transitions are slow because there is no API connection. This slows down the process. You can monitor these operations in the open browser.

Login password=secret123 and username=admin

**End points,**

**Singnaler to executor**

I create ,

Start\_Chart,Chart\_status,Chart\_Events,Chart\_closed\_event,debug\_executor,stop\_chart

->. This functions are listening signalr and also webhook. And writing response to receiver and fe live stream event screen

**Fe-to-executor-to-fe**

Generate\_event,Push\_event,Event\_stream

Function are working to fe and all event. They are listing all events and showing in the web page.