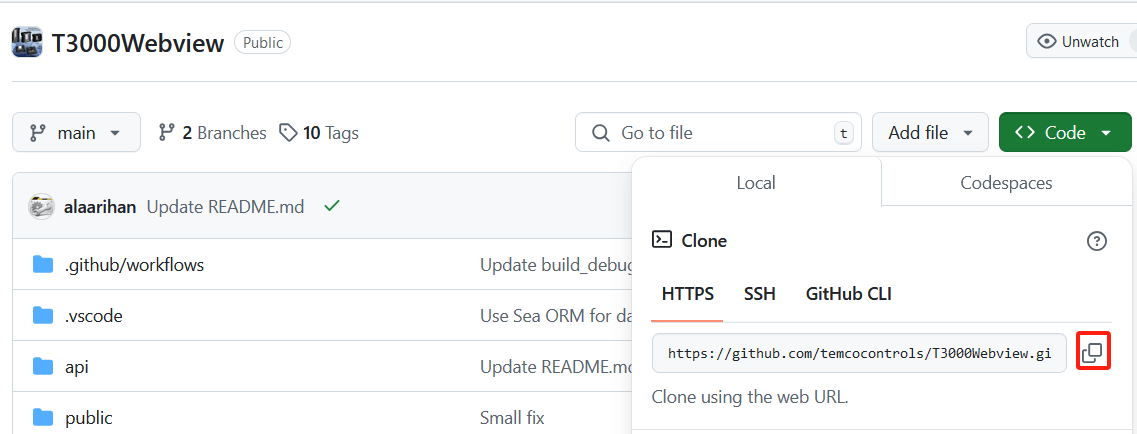
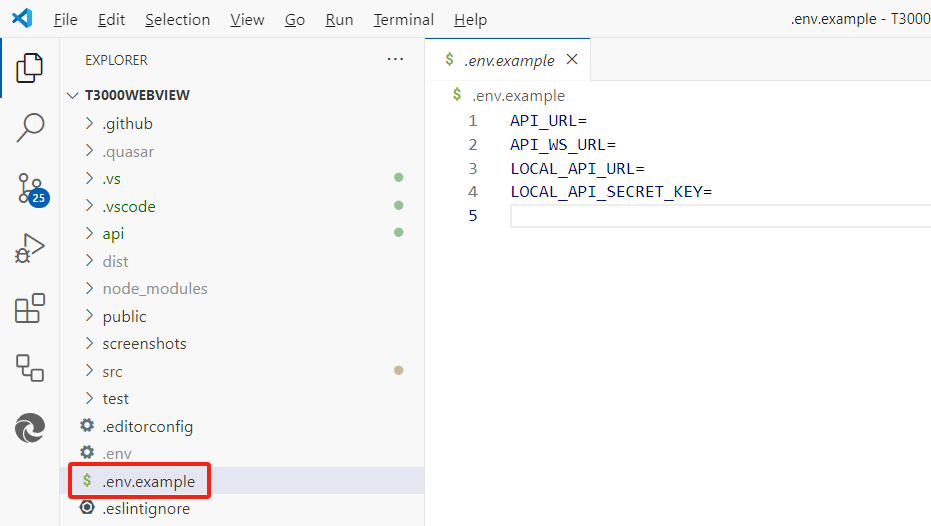
Debugging T3000Webview

1. clone the T3000Webview source code to local with below command.

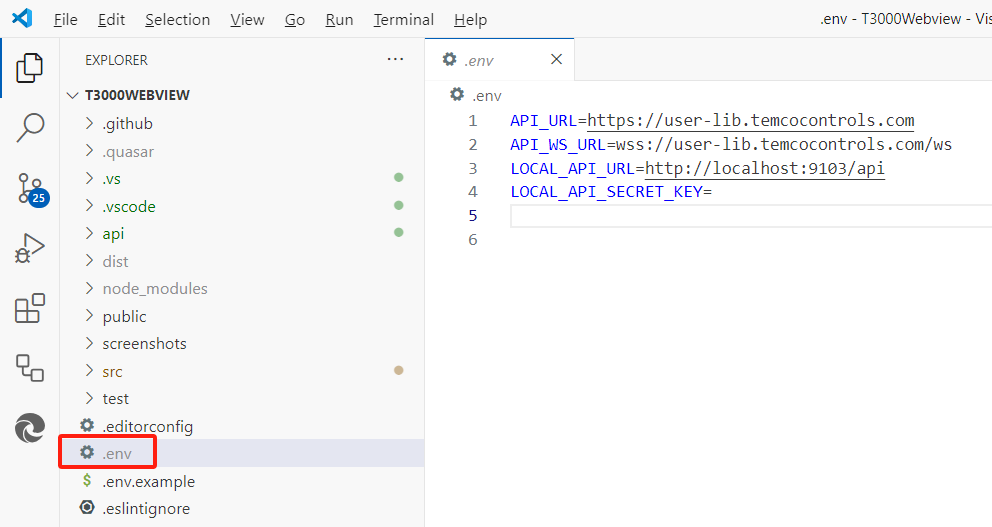
git clone <https://github.com/temcocontrols/T3000Webview.git>



2. open the source code via Visual Studio Code (or some other tool), find the .env.example file in the root folder “\T3000WEBVIEW”



3. add a new config file with name ”.env” and copy the keys from “.env.example” and set the new values as below.



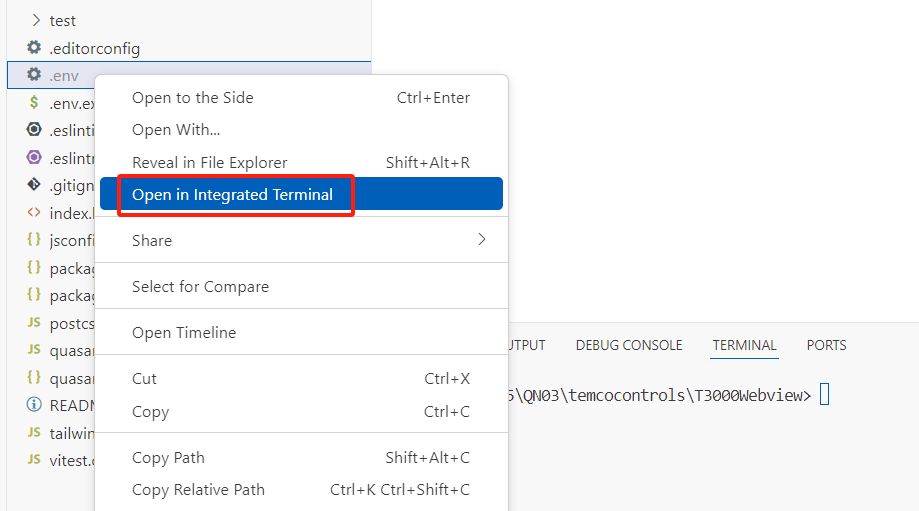
API\_URL=https://user-lib.temcocontrols.com

API\_WS\_URL=wss://user-lib.temcocontrols.com/ws

LOCAL\_API\_URL=http://localhost:9103/api

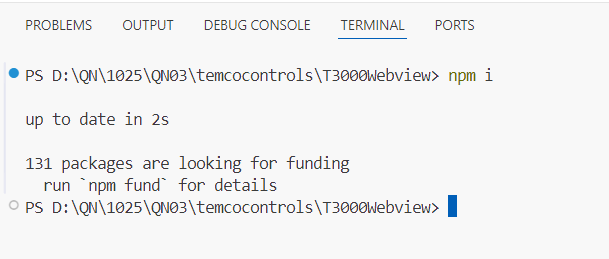
LOCAL\_API\_SECRET\_KEY=

4. right click to open the “Integrated Terminal”.



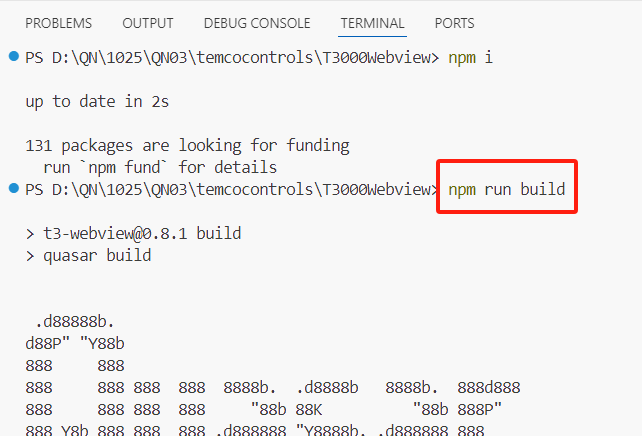
5. install the packages with below command.

npm i

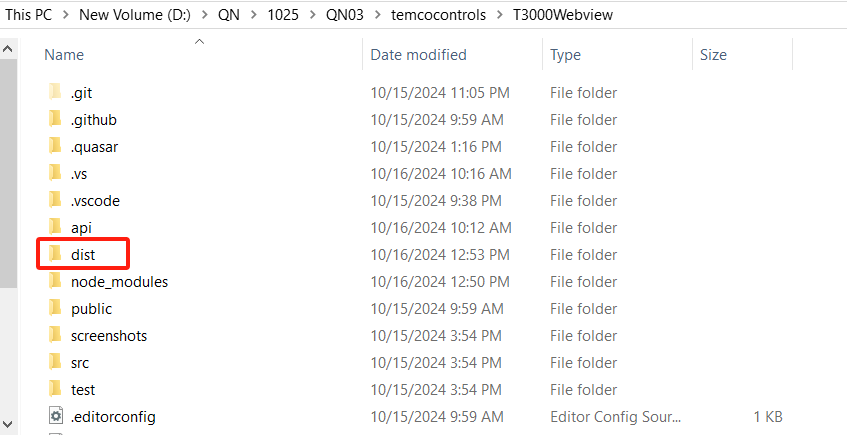


6. build the web project with below command.

npm run build

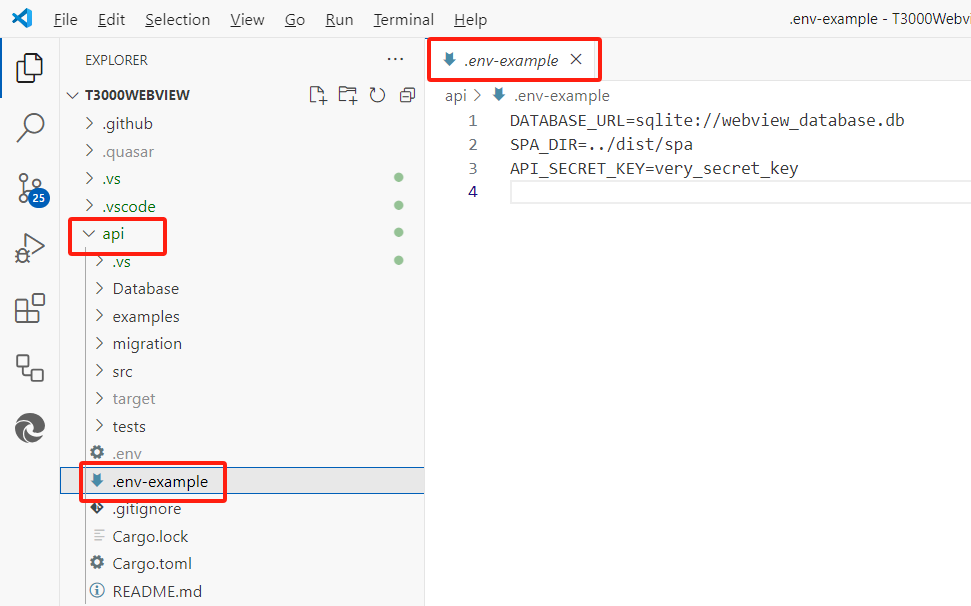




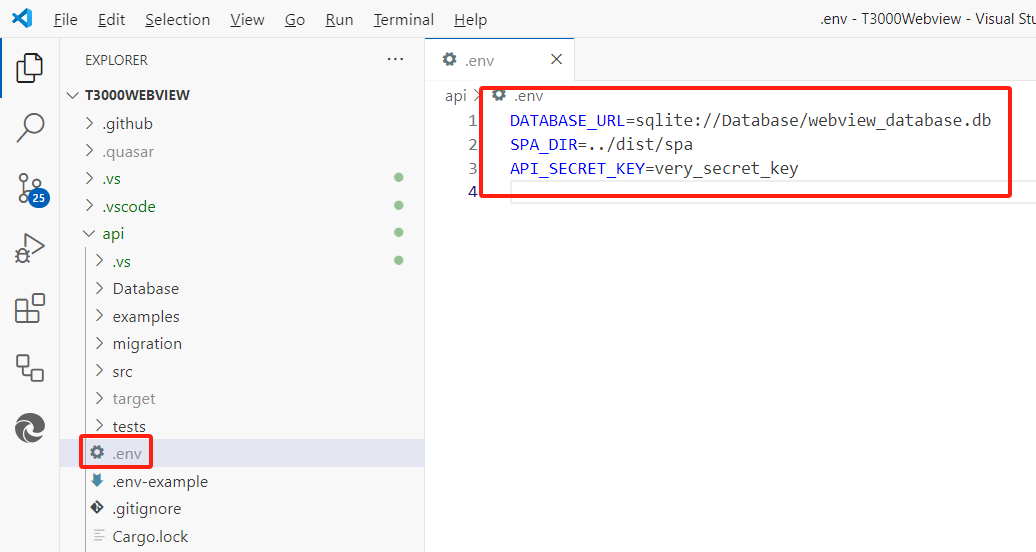


7. open the “.env-example” file under api folder. (full path should be \T3000Webview\api)

The api folder stores the source code of Rust back-end api



8. and a new config file under “\T3000Webview\api” with name “.env” and copy the keys from “.env-example”, update the values as below.



DATABASE\_URL=sqlite://Database/webview\_database.db

SPA\_DIR=../dist/spa

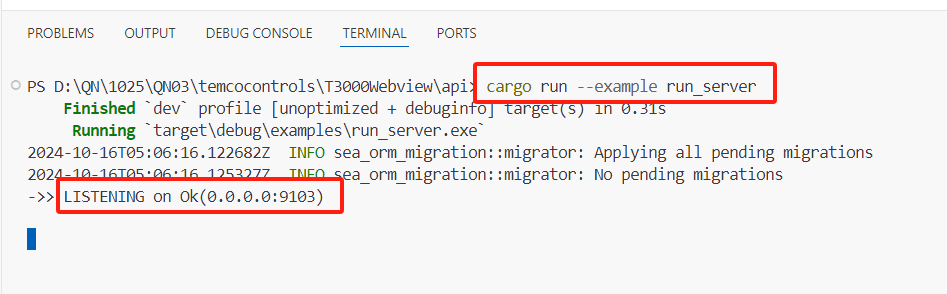
API\_SECRET\_KEY=very\_secret\_key

9. open the “Integrated Terminal” again, and cd into api folder



10. execute following command to start the rust api server.

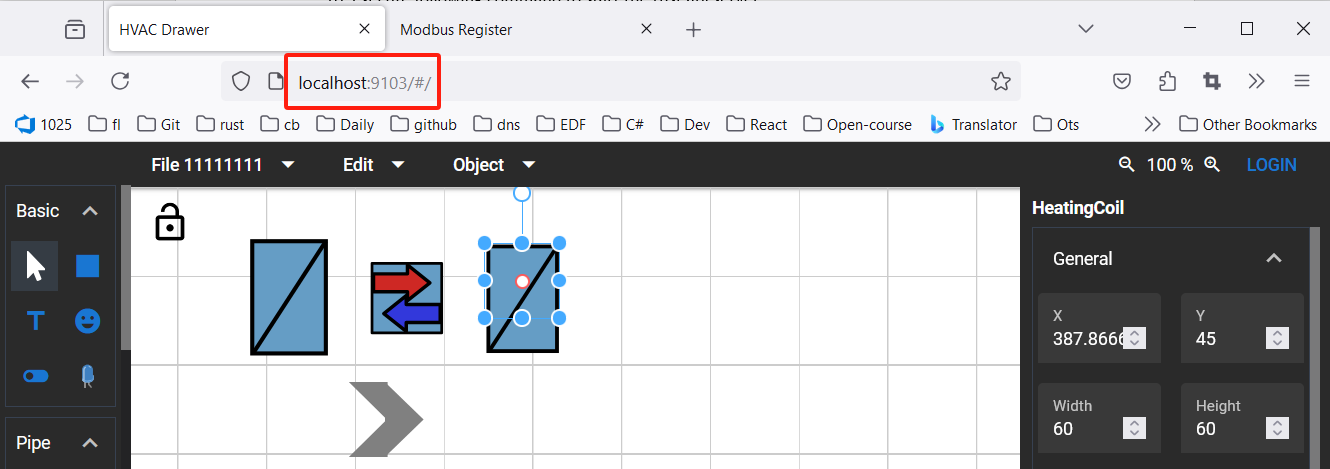
cargo run --example run\_server

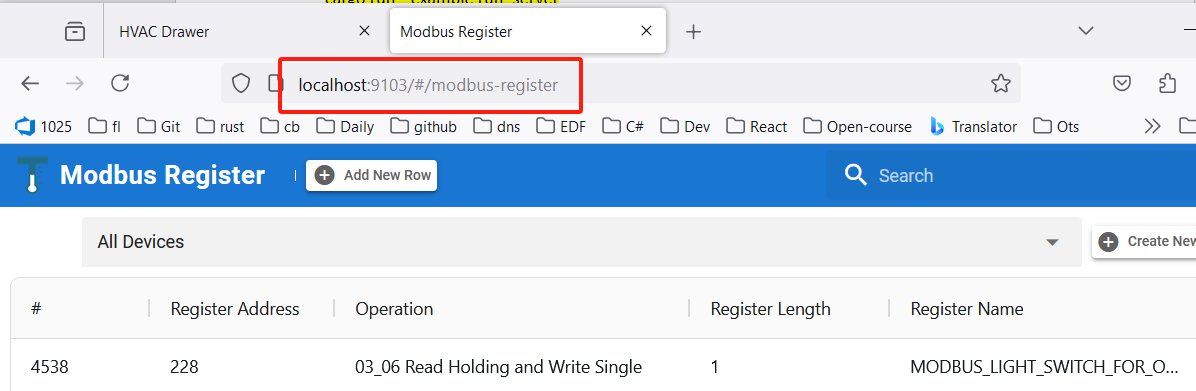


check with below urls to make sure the server has been started and working well.

<http://localhost:9103/#/>

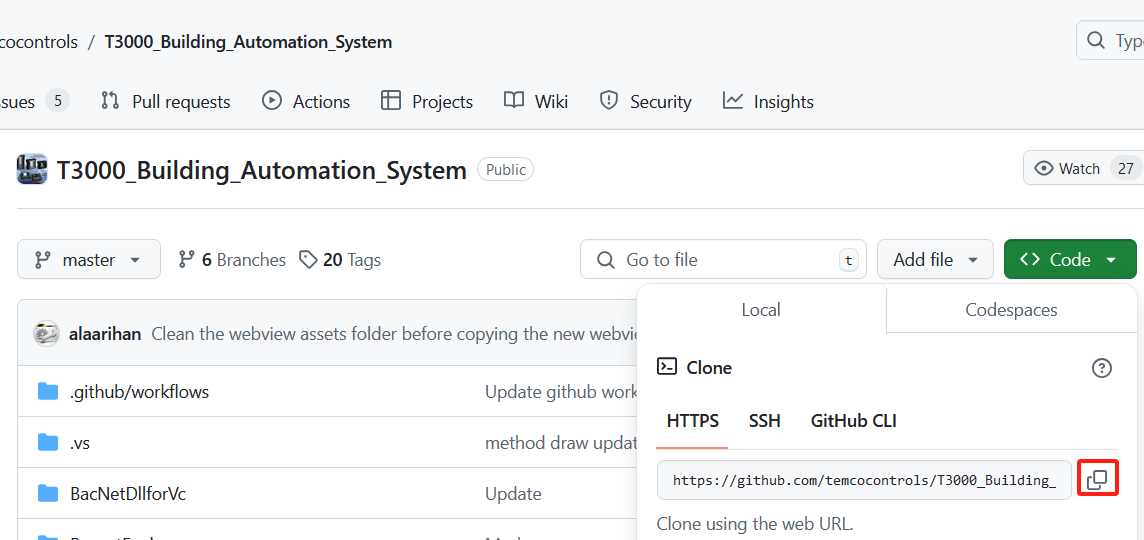
<http://localhost:9103/#/modbus-register>





Notes: if no need to debug it via T3000, can ignore below steps 11 & 12, can just use any web browser to do it.

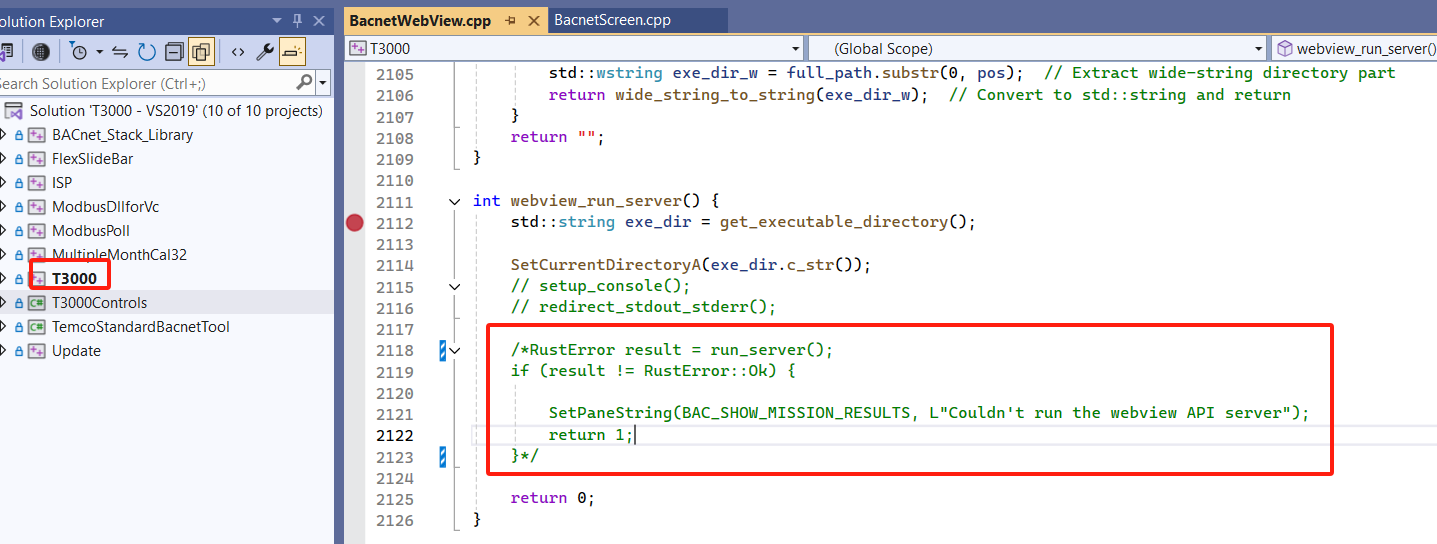
11. clone or update the source code of T3000\_Buiding\_Automation\_System, make sure it’s updated.



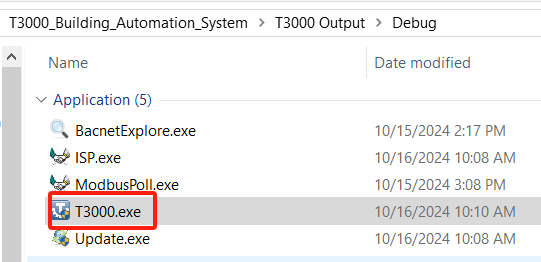
12. open the “T3000 - VS2019.sln” solution file with visual studio as administrator.

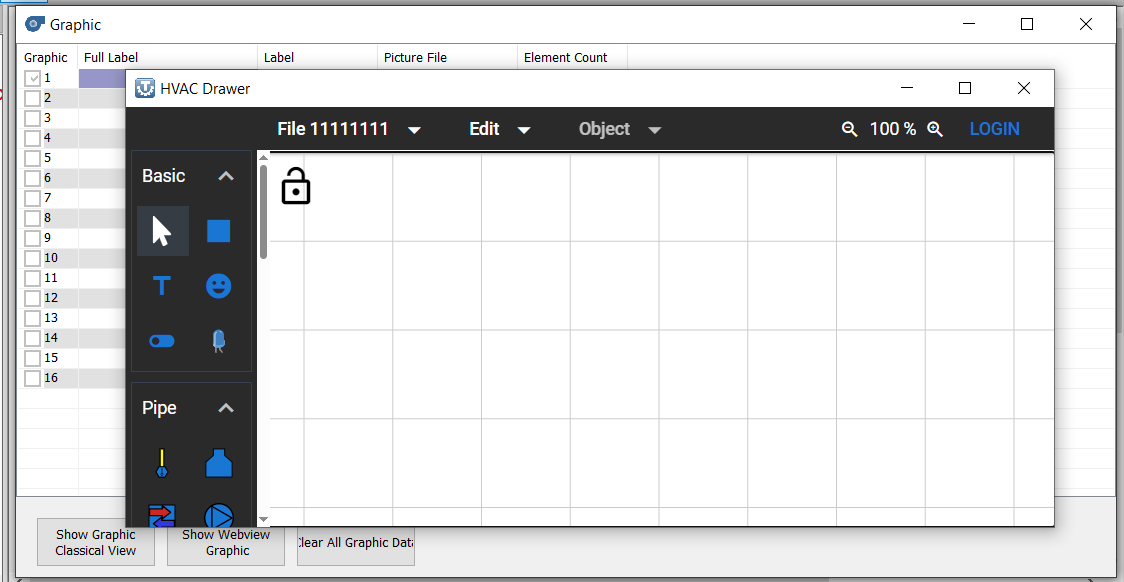
comment the code as shown below.

Make sure the T3000.exe does not start the rust server in background.

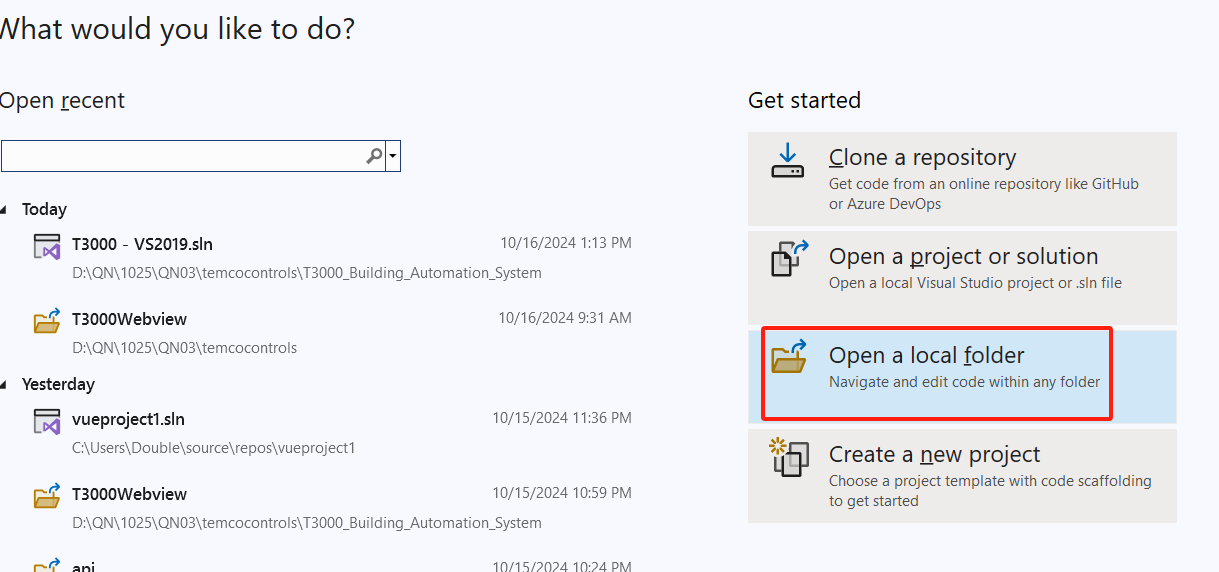


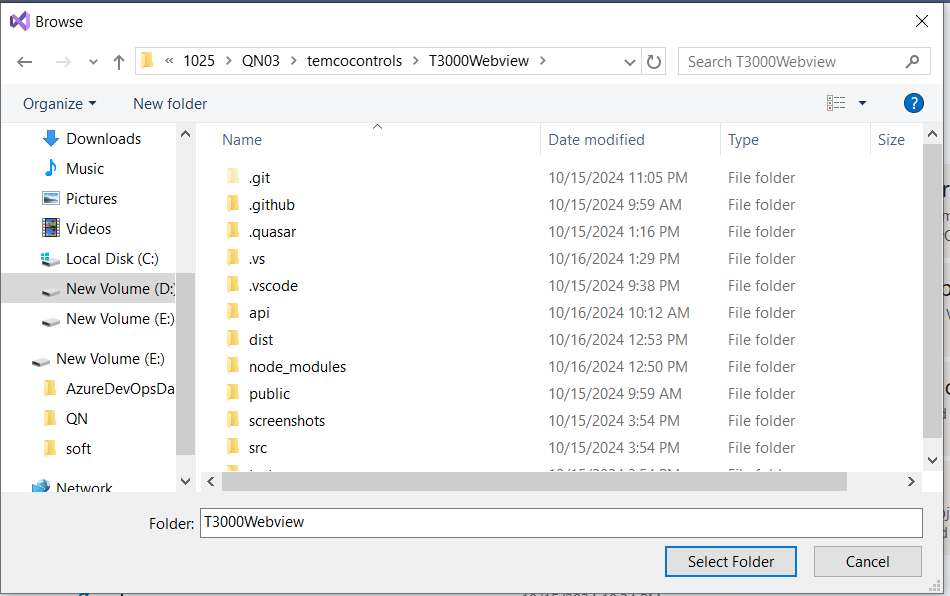
13. rebuild the solutions and run T3000.exe under T3000 Ouput\Debug, and check whether the “Show Webview Graphic” can access the rust server which was started by step 10



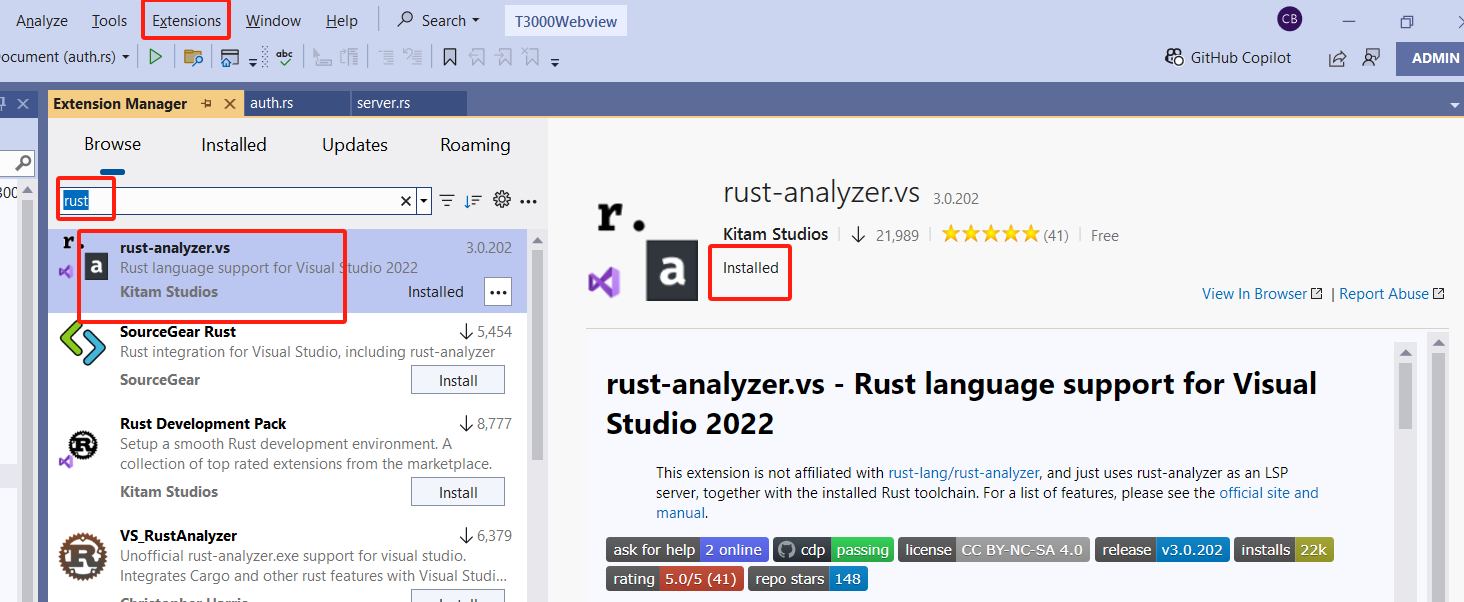


14. open the T3000Webview folder with Visual Studio 2022.

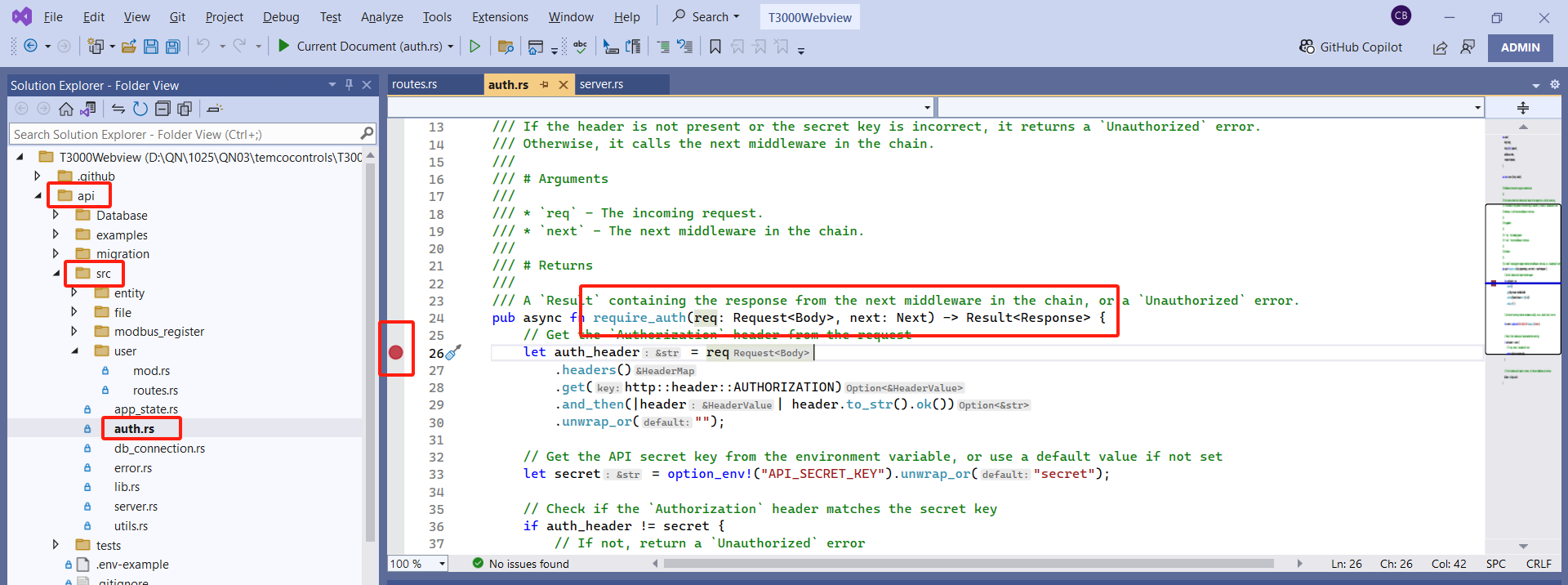




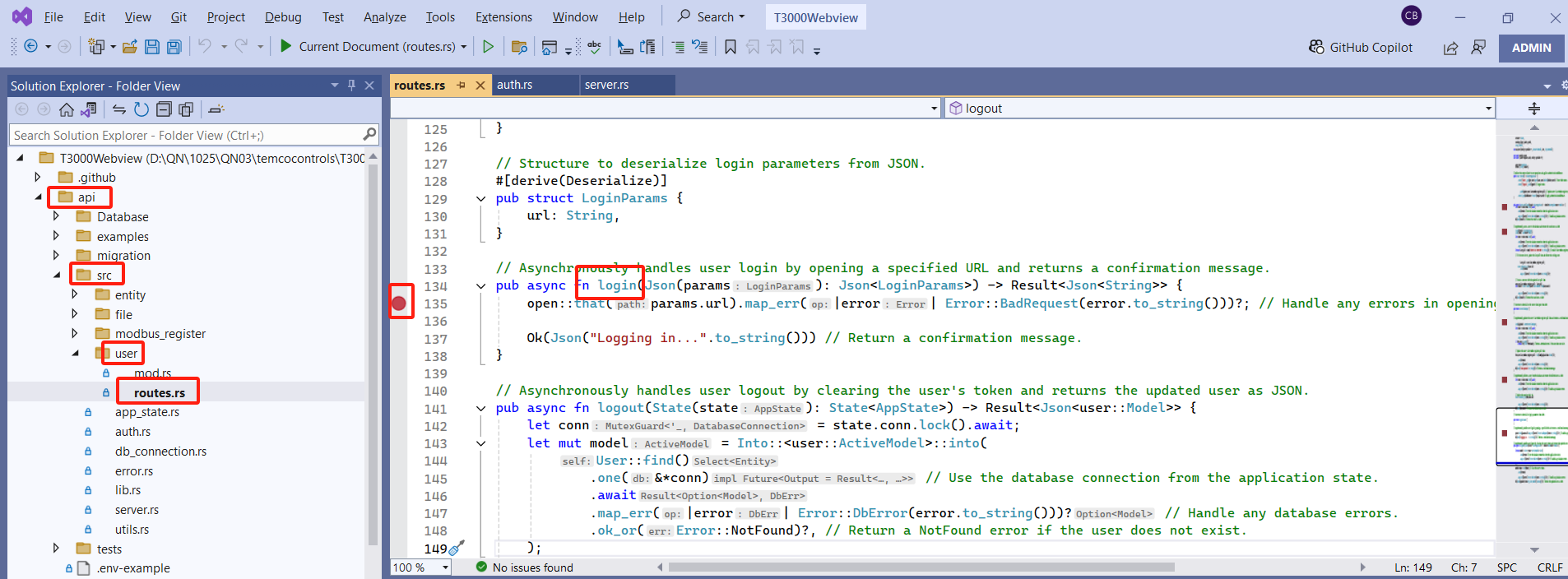
15. add rust-analyzer.vs extension to visual studio 2022.



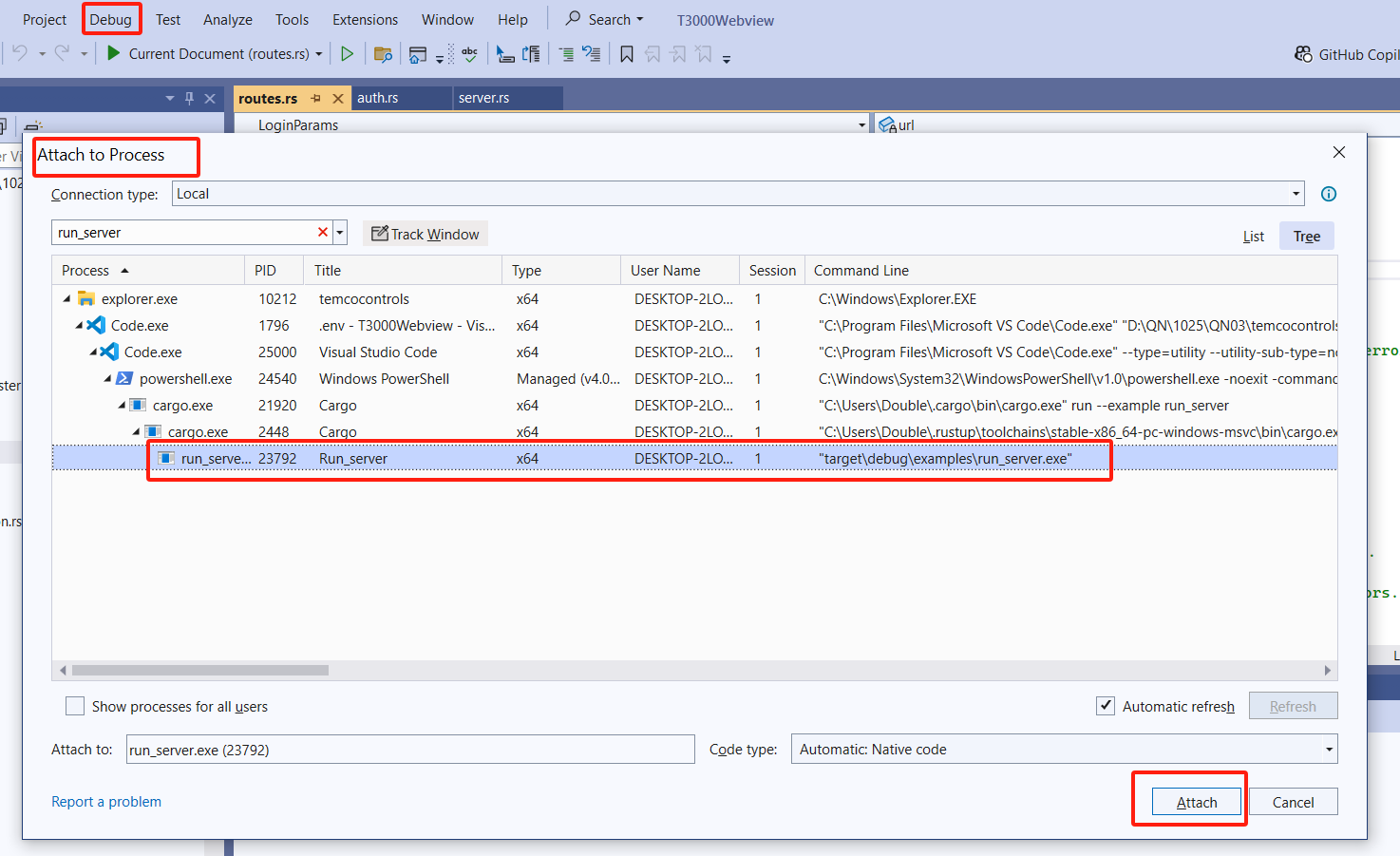
16. find auth.rs file under /api/src/auth.rs, and add a breakpoint for “require\_auth” function.



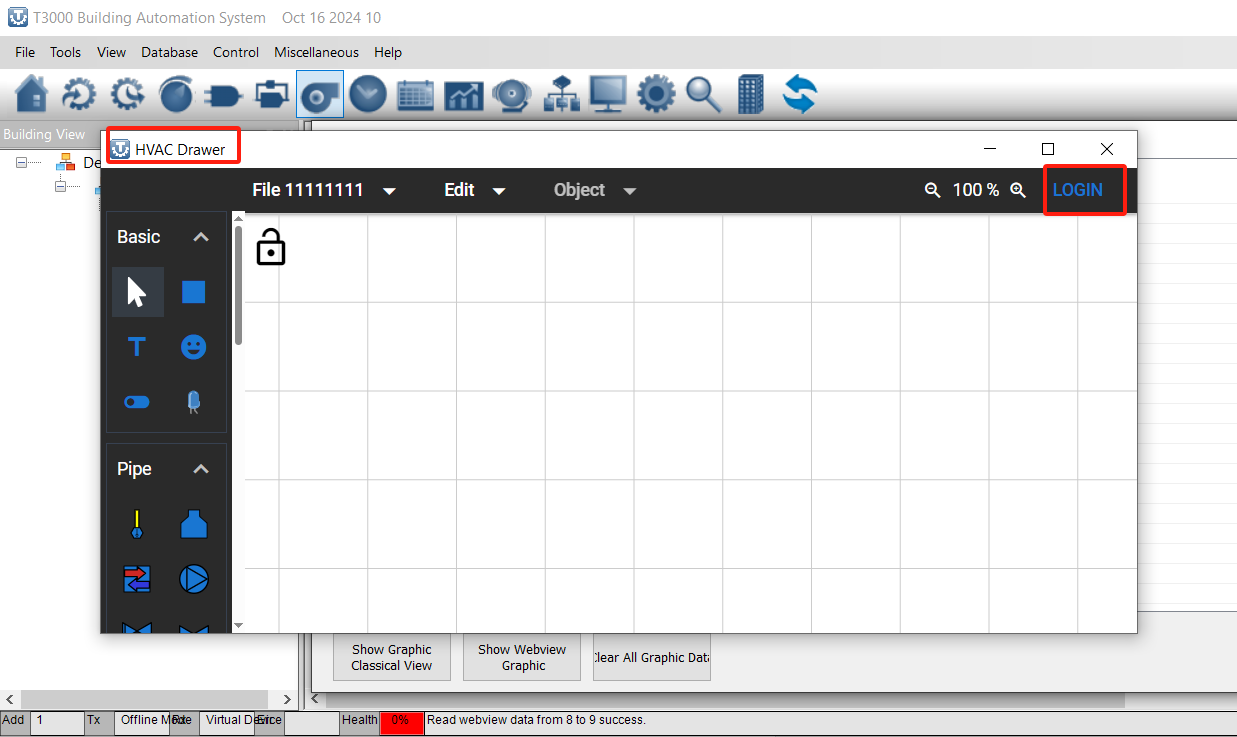
add another breakpoint for “login” function

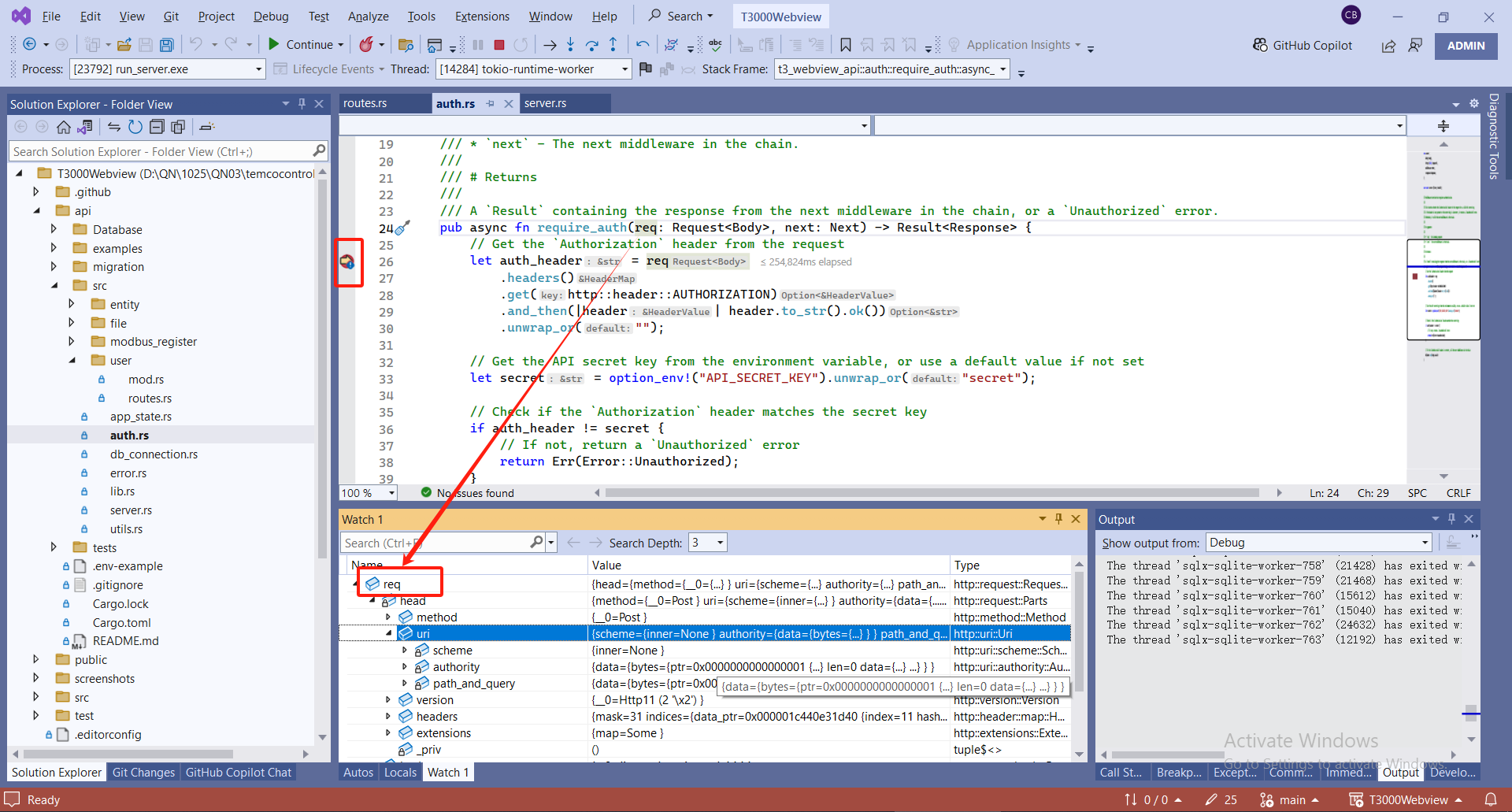


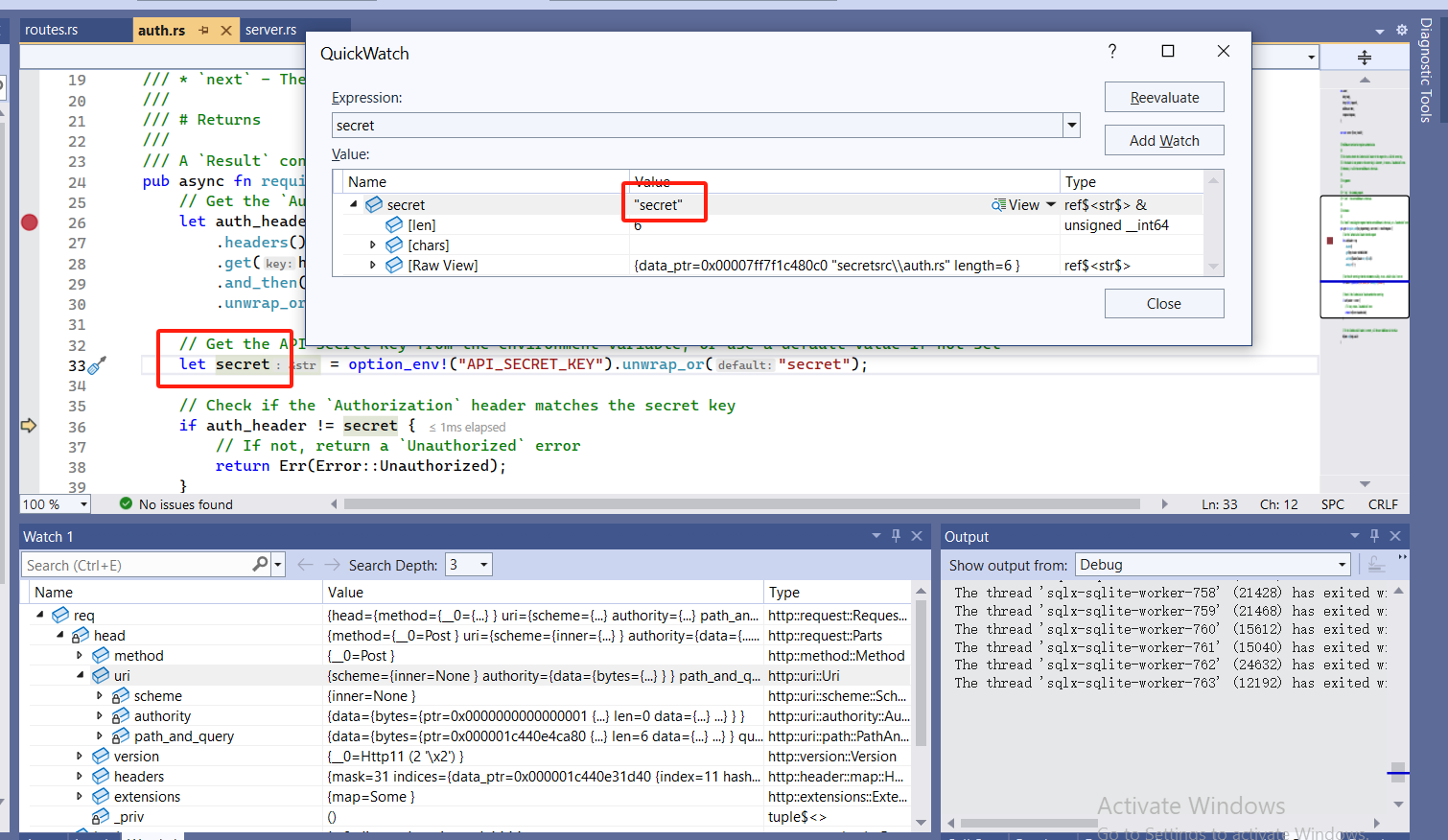
17. open the “Attach to Process” dialog, “Debug->Attach to Process”, and find the “run\_server” process to attach.



18. open “HVAC Drawer” and click the “LOGIN” button, it will redirect to the source code for debugging. You can check the details in visual studio.







19. for the JavaScript and Html files, can open the Edge Dev Tools to check the log and some other details.

