Table of content

Installation and uninstallation procedure. 2

Installation: 2

Uninstallation: 5

Configuration options and how to customize them. 7

Dependencies and any external libraries used. 9

Brief overview of the code structure and important modules. 11

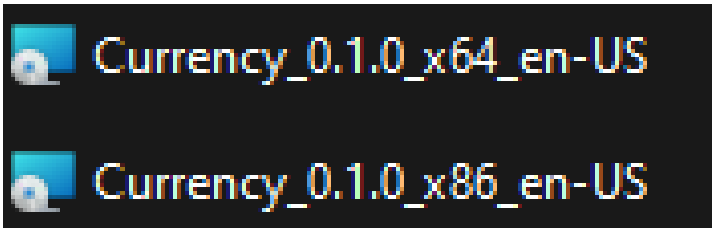
Troubleshooting tips and common issues. 12

Installation and uninstallation procedure.

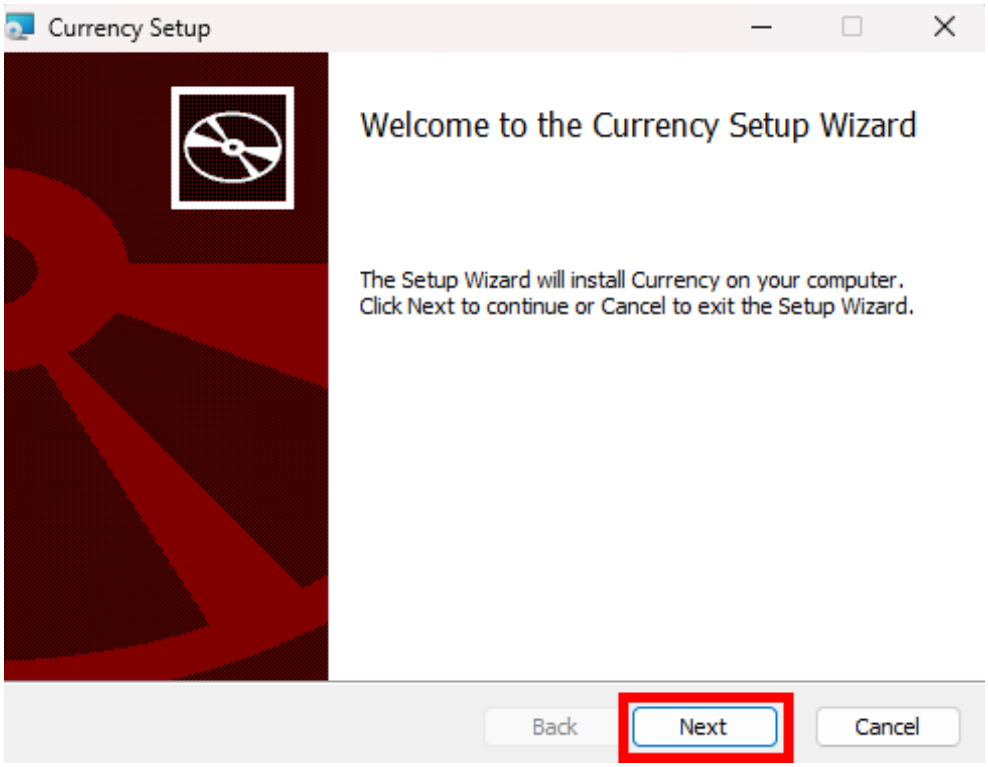
The app comes with both an installer and an uninstaller, making the installation and removal process similar to that of any other program.

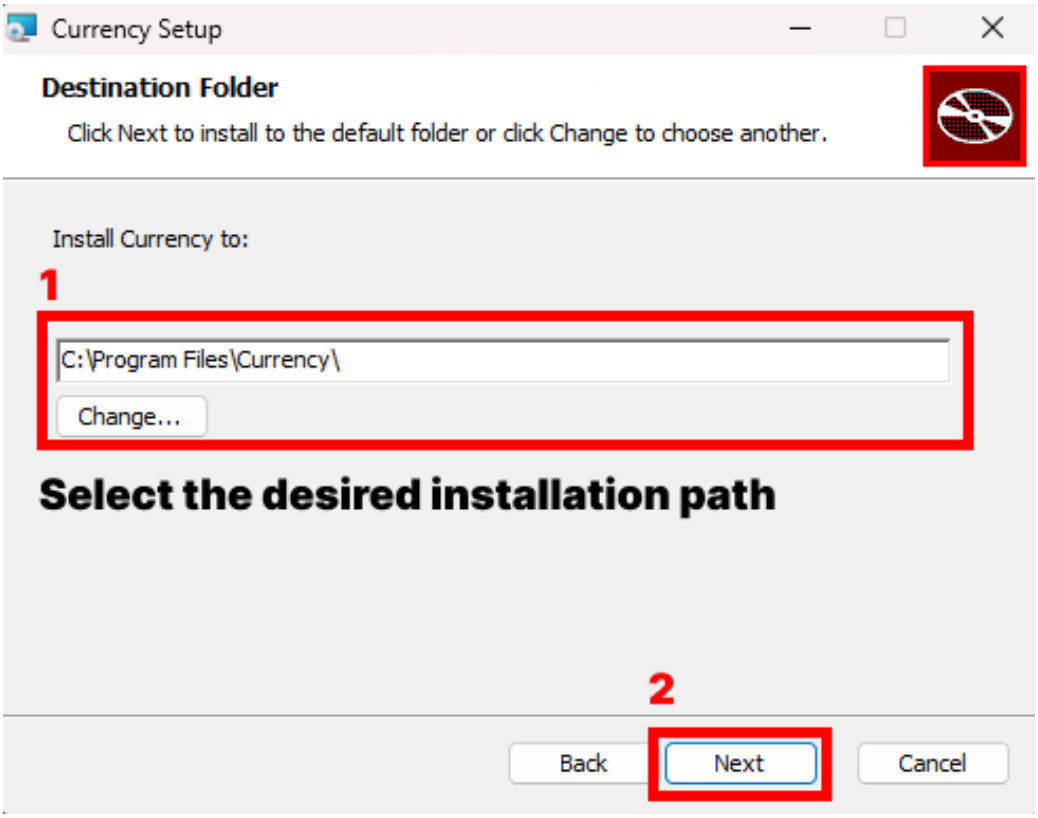
Installation:

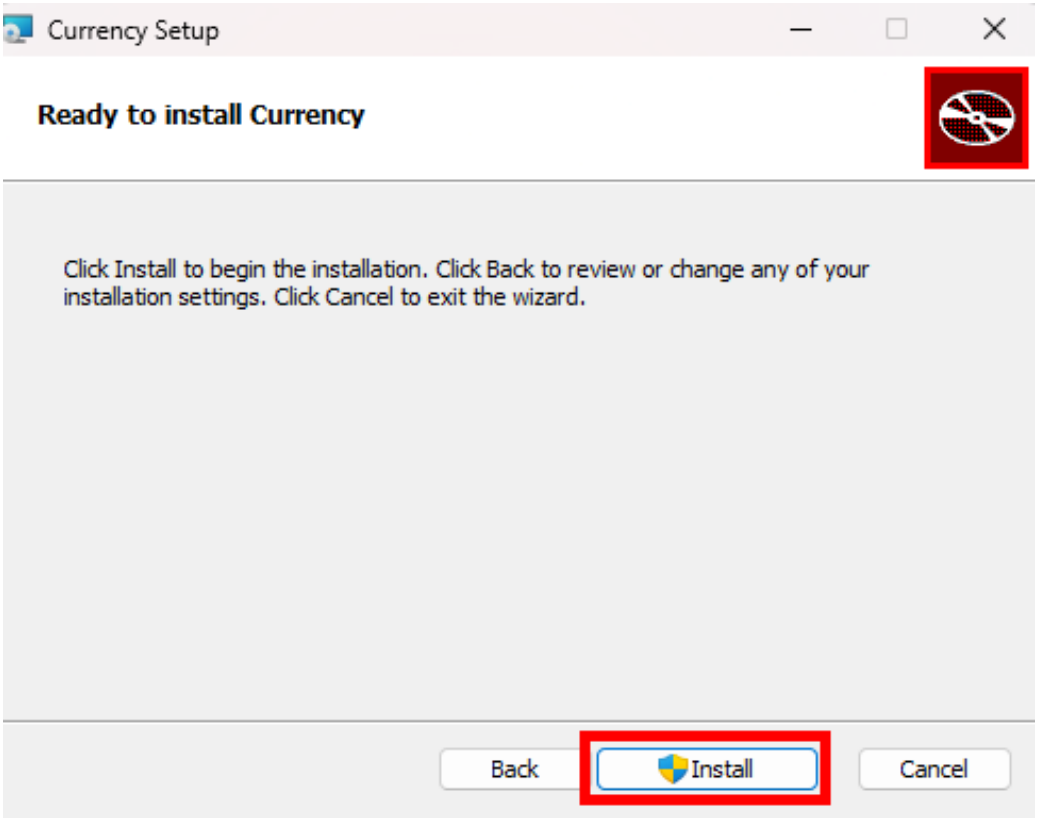
1. In the build folder, you can find installers for 32-bit Windows (Currency_0.1.0_x86_en-US) and 64-bit Windows (Currency_0.1.0_x64_en-US). We recommend using the MSI installer, although you have the option to use NSIS if you prefer.

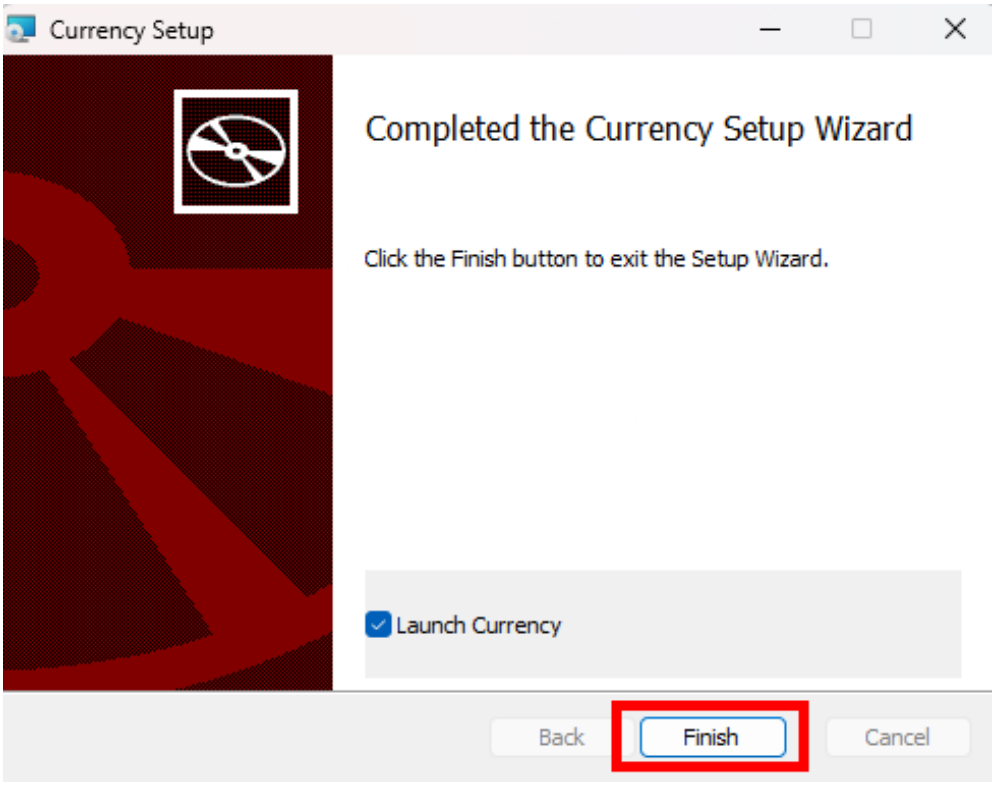


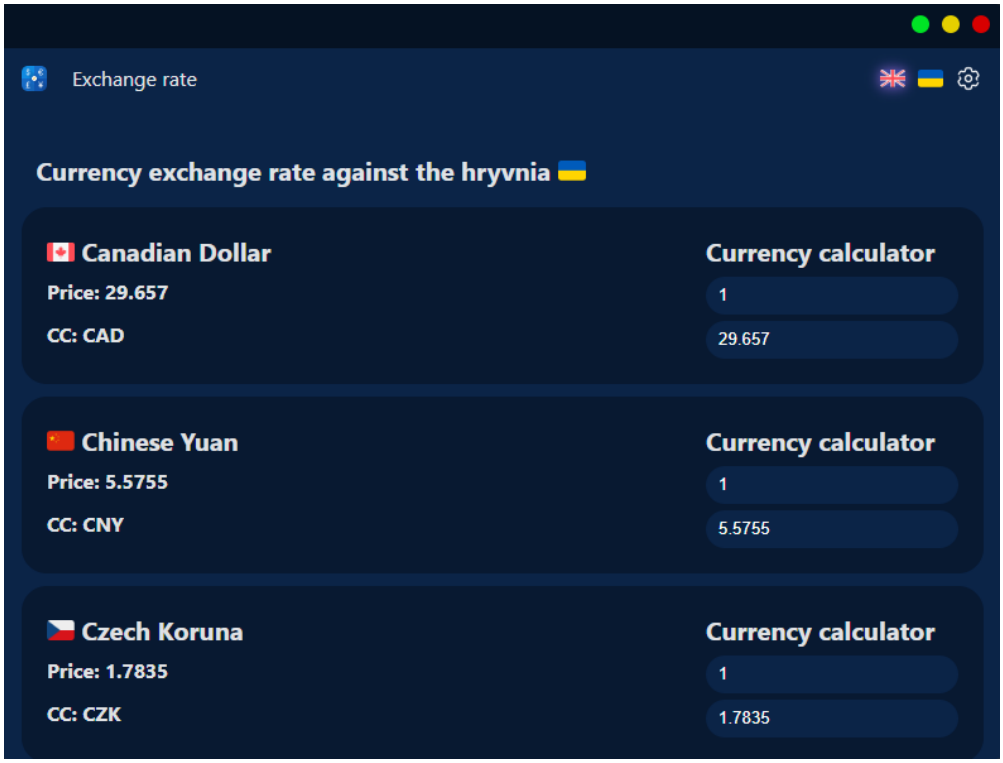
2. Open Currency_0.1.0_(x64/x86)_en-US and install the app as usual. Use the pictures below as a guide.




Currency Setup
Destination Folder
Click Next to install to the default folder or click Change to choose another.
Install Currency to:
1
C:\Program Files\Currency\
Change...
Select the desired installation path
2
Back
Next
Cancel


Currency Setup
Ready to install Currency
Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.
Back
Install
Cancel

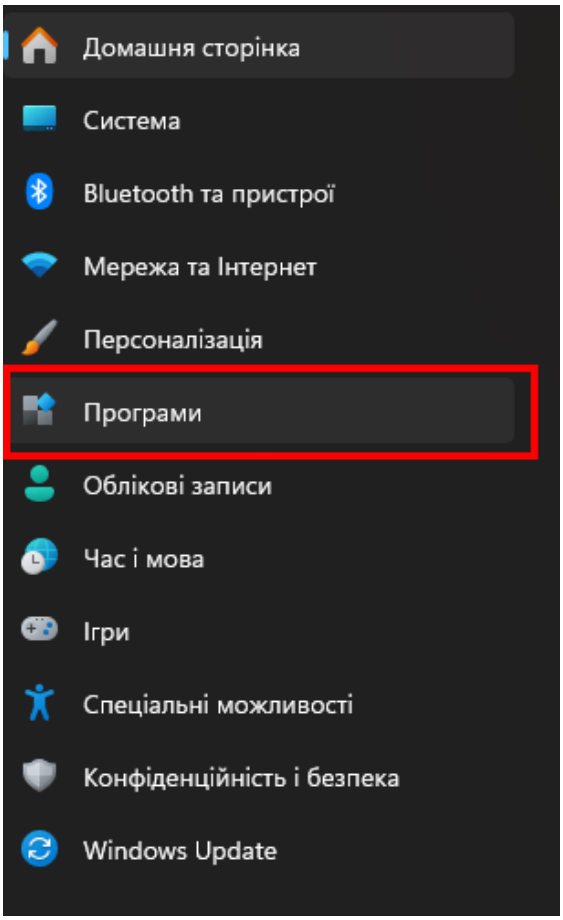

Currency Setup
Completed the Currency Setup Wizard
Click the Finish button to exit the Setup Wizard.
Launch Currency
Back
Finish
Cancel


Exchange rate
Currency exchange rate against the hryvnia
Canadian Dollar
Price: 29.657
CC: CAD
Currency calculator
1
29.657
Chinese Yuan
Price: 5.5755
CC: CNY
Currency calculator
1
5.5755
Czech Koruna
Price: 1.7835
CC: CZK
Currency calculator
1
1.7835

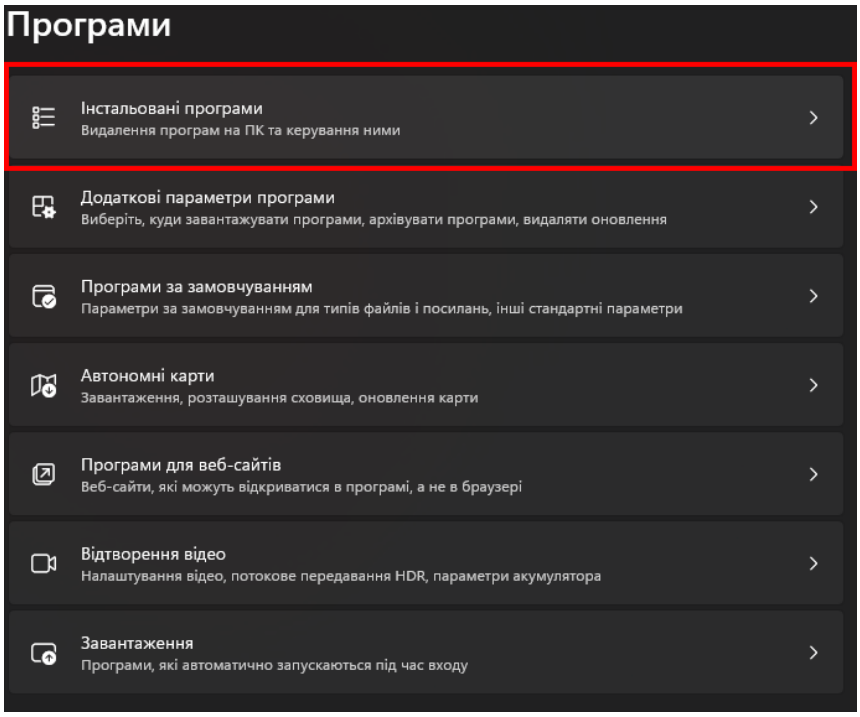
Uninstallation:

1. Press Win + I on keyboard

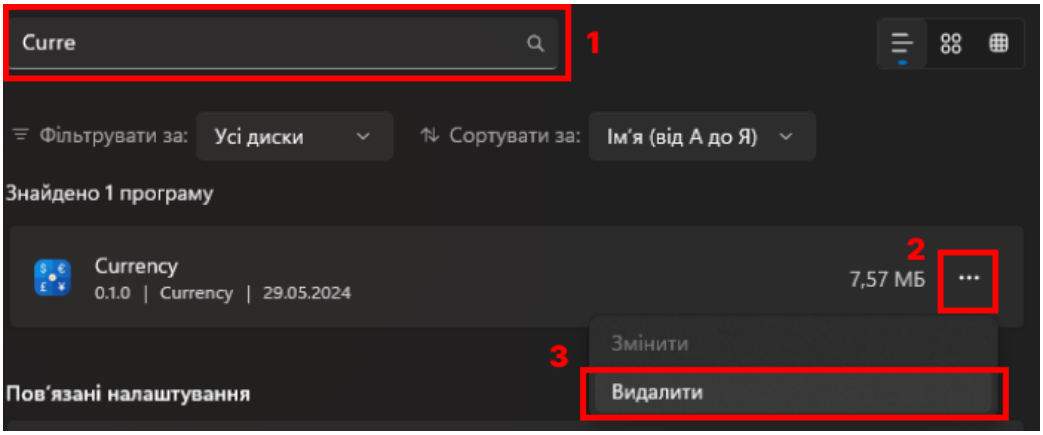
2. In left menu select “Programs”



3. Select “Installed programs”



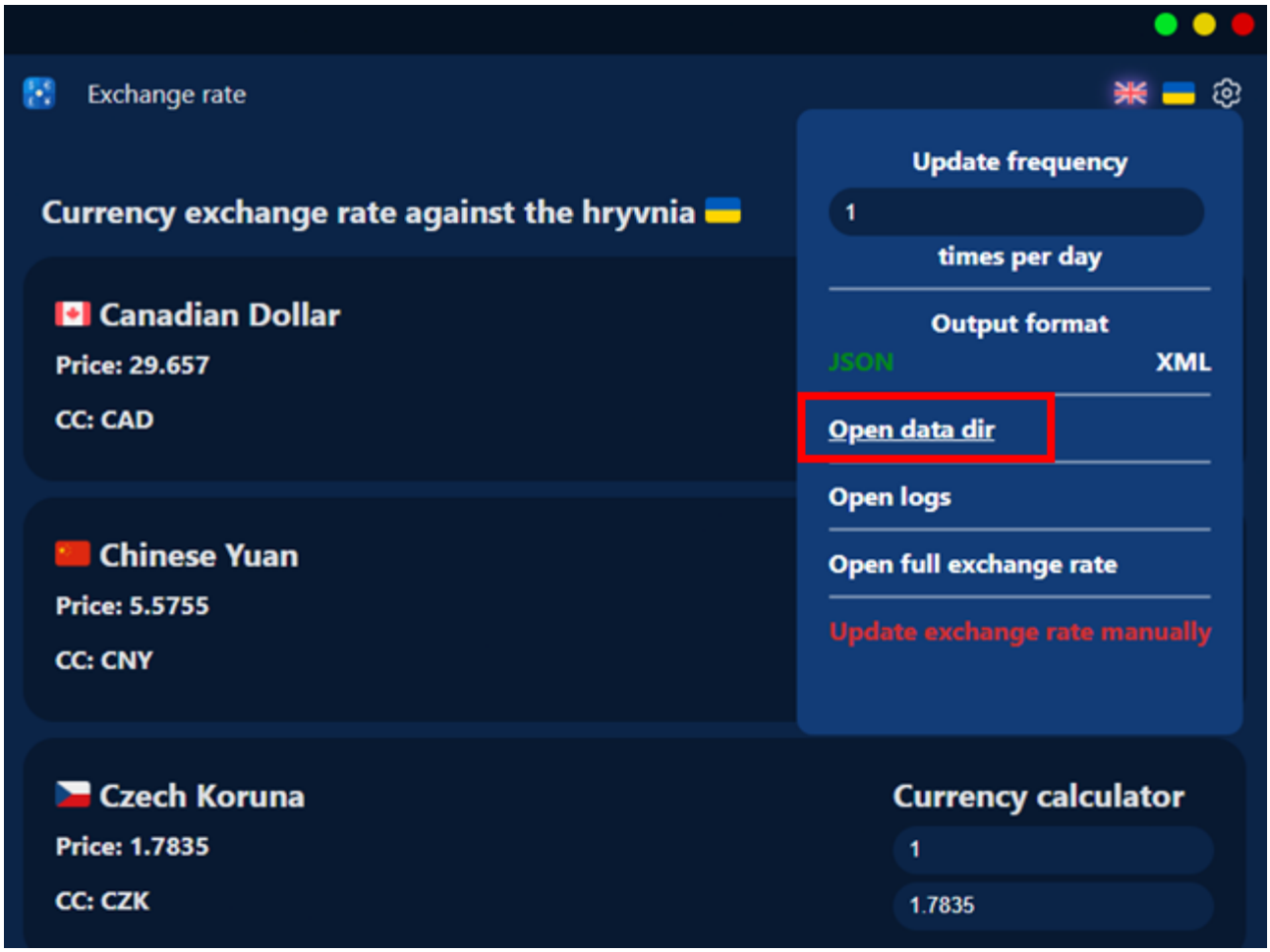
4. Use the filter to locate the app and delete it as shown in the picture.



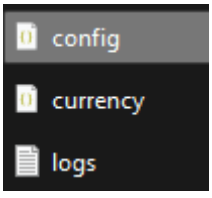
Configuration options and how to customize them.

Here is two ways to customize options:

1. **Hard method:** to access configuration file use settings panel in app and open data directory



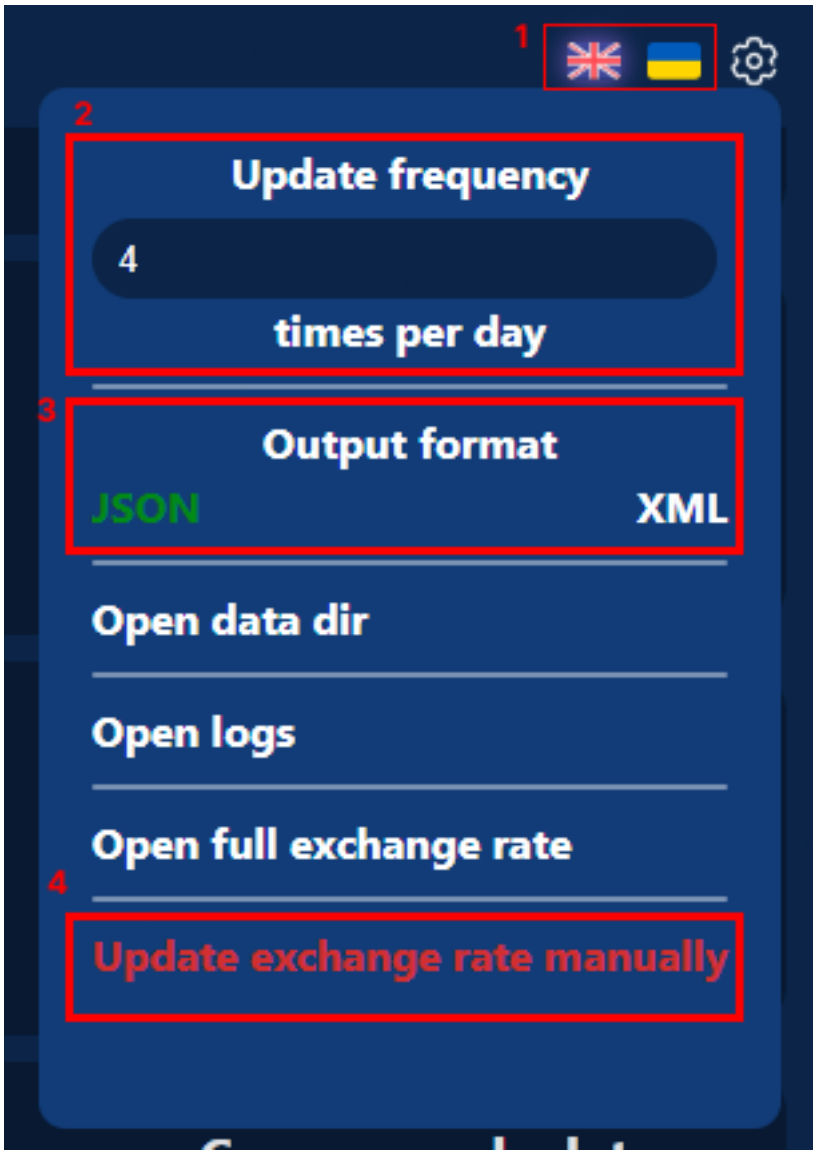
In opened folder select config.json*



Config file content with explanation showed down

```
{
  "frequency": "1", // How many times per day will update currency rates (min 1, max 4)
  "language": "en", // App language (you can select between en and uk)
  "output": "JSON", // Output format for currency rates (JSON or XML)
  "last_update": "1716938666" // Timestamp of last currency rates sync (DON'T CHANGE MANUALLY)
}
```

2. **Soft (normal):** to change configuration use GUI provided in the application



1 – Language configuration use between EN/UK

2 – How many times per day will update currency rates (select between 1 and 4 included)

3 – Output format for currency rates

4 – Update currency rates manually now instead of automated updates

1 – How many hryvnias to convert

2 – Converted value at the current exchange rate

Dependencies and any external libraries used.

serde_json

- **Purpose:** This library is used for serializing and deserializing data in JSON format.
- **Reason:** JSON is a common data interchange format, and serde_json provides an efficient and straightforward way to handle JSON data in Rust applications.

tauri

- **Purpose:** Tauri is a framework for building tiny, fast binaries for all major desktop platforms.
- **Reason:** The specified features enable various window management functionalities, allowing the application to control window behavior, position, and size across different operating systems, making it versatile for desktop application development.

window-shadows

- **Purpose:** This library is used to add drop shadows to application windows.
- **Reason:** Visual enhancements like window shadows improve the aesthetic appeal of desktop applications, providing a more polished and modern user interface.

reqwest

- **Purpose:** reqwest is a popular HTTP client library in Rust.
- **Reason:** It is used for making HTTP requests, which is essential for interacting with web services, APIs, and fetching data from the internet.

xml2json-rs

- **Purpose:** This library converts XML data to JSON format.
- **Reason:** Converting XML to JSON can be useful for applications that need to integrate with systems producing XML but prefer to work with JSON data internally, simplifying data handling and processing.

chrono

- **Purpose:** chrono is a date and time library for Rust.
- **Reason:** Handling dates and times is a common requirement in many applications, and chrono provides comprehensive functionality for dealing with time zones, formatting, and parsing dates and times efficiently.

National Bank of Ukraine (NBU) Open Data API

The National Bank of Ukraine (NBU) provides an Open Data API that allows developers to access various datasets related to financial and economic indicators of Ukraine. This API can be used to retrieve information about exchange rates, monetary statistics, and other financial data.

Brief overview of the code structure and important modules.

```
📦src - module with frontend tauri
┣📂assets - module with images
┣📂config - module with predefined files used to create start config and lang
┃ ┗ 📂language
┣ 📂lib - module with currency rates/exchange UI
┣ 📂scripts - module with TS scripts used in project
┣ 📂styles - module with TS styles used in project
📦src-tauri - module Rust code
┣📂src
┃  ┣ 📂config - module to work with config of project *1
┃  ┃  ┗ 📜mod.rs
┃  ┣ 📂log - module to work with logging *2
┃  ┃  ┗ 📜mod.rs
┃  ┣ 📂req - module to work with request to external APIs *3
┃  ┃  ┗ 📜mod.rs
┃  ┗ 📜main.rs - main rust file to build tauri application
┣ README.md
┗ README.PDF
```

*1 contains functions:

```
ensure_config_exist(config_path: &PathBuf, data_dir: &String) // Check if config file exist, if not create new

modify_config(value: String, id: String, data_dir: String) // Write new value to config, uses ensure_config_exist to check if config exist before writing value

read_config_value(key: String, data_dir: String) // Read value from config, uses ensure_config_exist to check if config exist before reading value
```

*2 contains functions:

```
log(level: String, msg: String, data_dir: String) // Write new logs to logs.txt file

open_log(data_dir: String) // Opens log file, used in UI “Open logs”
```

*3 contains functions:

```
get_currency_data(data_dir: String) // Reads data from currency.(JSON/XML) file and returns json to frontend

fetch_and_save_currency_rates(data_dir: String, manual: String) // Make request to external API to fetch currency rates then call save_currency_rates to save it


save_currency_rates(format: String,data_dir: &str, currency_rates: String) // Save currency rates to currency.(JSON/XML)
```

Troubleshooting tips and common issues.

On developing stage and tests was not recognized any issues 😊