CryptoBot — Starter Project

Нижче — повний початковий набір файлів для запуску бота + Dashboard + API + базова багатомовність.

Вміст розділений на файли. Скопіюй кожний файл у відповідний шлях у проекті `CryptoBot/`.

---

1) `README.md`

````
# 🚀 CryptoBot (Telegram + Dashboard)

## Вимоги
- Python 3.10+
- Рекомендовано: VS Code

## Інструкції
1. Клонуй або створи папку `CryptoBot` і поклади файли з цього документа.
2. Відкрий у VS Code.
3. Створи віртуальне середовище та активуй його:
   ```bash
   python3 -m venv venv
   source venv/bin/activate    # macOS/Linux
   venv\Scripts\activate     # Windows
   ```
4. Встанови залежності:
   ```bash
   pip install -r requirements.txt
   ```
5. Створи `.env` у корені проекту з таким вмістом:
   ```env
   BOT_TOKEN=your_telegram_token
   ADMIN_ID=your_telegram_id
   DASHBOARD_USER=admin
   DASHBOARD_PASS=changeme
   HOST=0.0.0.0
   PORT=8080
   ```
6. Запуск бота (polling):
   ```bash
   python3 bot.py
   ```
7. Запуск Dashboard:
   ```bash
   python3 dashboard.py
   ```

Dashboard буде доступний за адресою http://localhost:8080
````

```
---
```

---

2) `requirements.txt`

```
aiogram==2.25.1
Flask==2.2.5
python-dotenv==1.0.0
feedparser==6.0.10
tinydb==4.7.0
requests==2.31.0
```

---

3) `.env` (приклад — НЕ ЗАМІЩАЙ РЕАЛЬНИМИ КЛЮЧАМИ В ПУБЛІЧНИХ ЗОНАХ)

```
BOT_TOKEN=YOUR_TELEGRAM_BOT_TOKEN
ADMIN_ID=123456789
DASHBOARD_USER=admin
DASHBOARD_PASS=changeme
HOST=127.0.0.1
PORT=8080
```

---

4) `config.py`

```python
import os
from dotenv import load_dotenv

load_dotenv()

BOT_TOKEN = os.getenv("BOT_TOKEN")
ADMIN_ID = int(os.getenv("ADMIN_ID")) if os.getenv("ADMIN_ID") else None
DASHBOARD_USER = os.getenv("DASHBOARD_USER", "admin")
DASHBOARD_PASS = os.getenv("DASHBOARD_PASS", "changeme")
HOST = os.getenv("HOST", "127.0.0.1")
PORT = int(os.getenv("PORT", 8080))

# Paths
BASE_DIR = os.path.dirname(__file__)
DATA_DIR = os.path.join(BASE_DIR, "data")
if not os.path.exists(DATA_DIR):
    os.makedirs(DATA_DIR, exist_ok=True)
```

```
USERS_DB = os.path.join(DATA_DIR, "users.json")
CACHE_FILE = os.path.join(DATA_DIR, "cache.json")
```

---

5) core/database.py

```
from tinydb import TinyDB, Query
from config import USERS_DB

db = TinyDB(USERS_DB)
users = db.table('users')
alerts = db.table('alerts')
settings = db.table('settings')

def get_user(user_id):
    User = Query()
    res = users.get(User.id == user_id)
    return res

def upsert_user(user_id, data: dict):
    User = Query()
    if users.contains(User.id == user_id):
        users.update(data, User.id == user_id)
    else:
        users.insert({**{"id": user_id}, **data})

def list_users():
    return users.all()

# alerts: {id, user_id, symbol, target, active}

def add_alert(a):
    return alerts.insert(a)

def list_alerts(active_only=True):
    if active_only:
        return [a for a in alerts.all() if a.get('active', True)]
    return alerts.all()
```

---

6) core/language.py

```
import json
from pathlib import Path

TRANSLATIONS = {}
```

```
def load_translations():
    base = Path(__file__).parents[0] / 'translations'
    for p in base.glob('*.json'):
        TRANSLATIONS[p.stem] = json.loads(p.read_text(encoding='utf-8'))

def t(lang: str, key: str):
    if not TRANSLATIONS:
        load_translations()
    return TRANSLATIONS.get(lang, TRANSLATIONS.get('ua', {})).get(key, key)
```

---

7) `translations/ua.json`

```
{
  "welcome": "Вітаю! Оберіть опцію:",
  "choose_lang": "Оберіть мову / Choose language",
  "btn_converter": "💱 Конвертер",
  "btn_p2p": "   P2P Продавці",
  "btn_alerts": "🩶 Нагадування",
  "btn_advisor": "   Радник",
  "btn_forex": "💱 Валютний ринок",
  "btn_analytics": "🆚 Аналітика ринку",
  "btn_exchanges": "🌐 Моніторинг бірж",
  "btn_partners": "   Партнери",
  "btn_news": "💬 Новини",
  "btn_help": "ℹ️ Допомога"
}
```

---

8) `translations/en.json`

```
{
  "welcome": "Welcome! Choose an option:",
  "choose_lang": "Оберіть мову / Choose language",
  "btn_converter": "💱 Converter",
  "btn_p2p": "   P2P Sellers",
  "btn_alerts": "🩶 Reminders",
  "btn_advisor": "   Advisor",
  "btn_forex": "💱 Forex Market",
  "btn_analytics": "🆚 Market Analytics",
  "btn_exchanges": "🌐 Exchange Monitor",
  "btn_partners": "   Partners",
  "btn_news": "💬 News",
  "btn_help": "ℹ️ Help"
}
```

9) `bot.py` (головний, polling)

```
import logging
from aiogram import Bot, Dispatcher, types
from aiogram.utils import executor
from aiogram.types import ReplyKeyboardMarkup, KeyboardButton

from config import BOT_TOKEN
from core.language import t
from core import database

logging.basicConfig(level=logging.INFO)

bot = Bot(token=BOT_TOKEN)
dp = Dispatcher(bot)

# Simple in-memory default lang fallback
DEFAULT_LANG = 'ua'

@dp.message_handler(commands=['start'])
async def cmd_start(message: types.Message):
    # set default language for new users
    user = database.get_user(message.from_user.id)
    if not user:
        database.upsert_user(message.from_user.id, {'lang': DEFAULT_LANG})
    # Show language keyboard
    kb = ReplyKeyboardMarkup(resize_keyboard=True)
    kb.add(KeyboardButton('♀ Українська'), KeyboardButton('🔊 English'))
    await message.answer(t('ua', 'choose_lang'), reply_markup=kb)

@dp.message_handler(lambda m: m.text in ['♀ Українська', '🔊 English'])
async def set_lang(message: types.Message):
    lang = 'ua' if 'Укра' in message.text else 'en'
    database.upsert_user(message.from_user.id, {'lang': lang})
    await show_main_menu(message, lang)

async def show_main_menu(message: types.Message, lang: str):
    kb = ReplyKeyboardMarkup(resize_keyboard=True)
    kb.add(KeyboardButton(t(lang, 'btn_converter')))
    kb.add(KeyboardButton(t(lang, 'btn_p2p')))
    kb.add(KeyboardButton(t(lang, 'btn_alerts')))
    kb.add(KeyboardButton(t(lang, 'btn_advisor')))
    kb.add(KeyboardButton(t(lang, 'btn_forex')))
    kb.add(KeyboardButton(t(lang, 'btn_analytics')))
    kb.add(KeyboardButton(t(lang, 'btn_exchanges')))
    kb.add(KeyboardButton(t(lang, 'btn_partners')))
    kb.add(KeyboardButton(t(lang, 'btn_news')))
    kb.add(KeyboardButton(t(lang, 'btn_help')))
    await message.answer(t(lang, 'welcome'), reply_markup=kb)
```

```
# Handlers for menus (simple stubs)
@dp.message_handler(lambda m: m.text and 'Конвертер' in m.text or 'Converter'
in m.text)
async def handle_converter(message: types.Message):
    user = database.get_user(message.from_user.id) or {}
    lang = user.get('lang', DEFAULT_LANG)
    await message.answer(' 9 ' + t(lang, 'btn_converter') + '\n\nВведи в
форматі: 100 USD to UAH')

@dp.message_handler(lambda m: m.text and ('P2P' in m.text or 'Продавці' in
m.text or 'Sellers' in m.text))
async def handle_p2p(message: types.Message):
    user = database.get_user(message.from_user.id) or {}
    lang = user.get('lang', DEFAULT_LANG)
    # stub: return example partners
    text = '☯️ P2P Partners (example):\n1) Binance P2P - USDT\n2) Local
Seller - UAH'
    await message.answer(text)

@dp.message_handler(lambda m: m.text and ('Нагадування' in m.text or
'Reminders' in m.text))
async def handle_alerts(message: types.Message):
    await message.answer('🖤 Тут можна додавати/видаляти нагадування (поки
stub)')

@dp.message_handler(lambda m: m.text and ('Радник' in m.text or 'Advisor' in
m.text))
async def handle_advisor(message: types.Message):
    await message.answer('   Радник: короткі поради на основі даних (stub)')

@dp.message_handler(lambda m: m.text and ('Валютний ринок' in m.text or
'Forex' in m.text))
async def handle_forex(message: types.Message):
    await message.answer('💱 Валютний ринок: середній курс USD/UAH 37.00
(приклад)')

@dp.message_handler(lambda m: m.text and ('Аналітика' in m.text or
'Analytics' in m.text))
async def handle_analytics(message: types.Message):
    await message.answer('🆚 Аналітика: топ-5 змін за 24h (stub)')

@dp.message_handler(lambda m: m.text and ('Моніторинг бірж' in m.text or
'Exchange' in m.text))
async def handle_exchanges(message: types.Message):
    await message.answer('🌐 Моніторинг бірж: таблиця (відкриється у
Dashboard)')

@dp.message_handler(lambda m: m.text and ('Партнери' in m.text or 'Partners'
in m.text))
async def handle_partners(message: types.Message):
```

```
    await message.answer('   Партнери: Binance, Bybit (реф. посилання у
Dashboard)')

@dp.message_handler(lambda m: m.text and ('Новини' in m.text or 'News' in
m.text))
async def handle_news(message: types.Message):
    await message.answer('💬 Новини: отримати останні новини (stub)')

@dp.message_handler(lambda m: m.text and ('Допомога' in m.text or 'Help' in
m.text))
async def handle_help(message: types.Message):
    await message.answer('ℹ️ FAQ: /start щоб повернутися до меню')

if __name__ == '__main__':
    executor.start_polling(dp, skip_updates=True)
```

---

10) `dashboard.py` (Flask simple Dashboard + API)

```
from flask import Flask, render_template, request, redirect, url_for,
session, jsonify
from config import DASHBOARD_USER, DASHBOARD_PASS, HOST, PORT
from core import database
import os

app = Flask(__name__)
app.secret_key = os.urandom(24)

@app.route('/')
def index():
    if not session.get('logged_in'):
        return redirect(url_for('login'))
    # simple dashboard data
    users = database.list_users()
    alerts = database.list_alerts(active_only=False)
    return render_template('dashboard.html', users=users, alerts=alerts)

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        user = request.form.get('username')
        pw = request.form.get('password')
        if user == DASHBOARD_USER and pw == DASHBOARD_PASS:
            session['logged_in'] = True
            return redirect(url_for('index'))
        return 'Invalid creds', 401
    return render_template('login.html')

@app.route('/logout')
```

```
def logout():
    session.clear()
    return redirect(url_for('login'))

# API endpoints
@app.route('/api/stats')
def api_stats():
    return jsonify({
        'users': len(database.list_users()),
        'alerts': len(database.list_alerts(active_only=False))
    })

@app.route('/api/alerts')
def api_alerts():
    return jsonify(database.list_alerts(active_only=False))

if __name__ == '__main__':
    app.run(host=HOST, port=PORT, debug=True)
```

---

11) `web/templates/base.html`

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>CryptoBot Dashboard</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/
bootstrap.min.css" rel="stylesheet">
  </head>
  <body>
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
      <div class="container-fluid">
        <a class="navbar-brand" href="#">CryptoBot Dashboard</a>
        <div class="d-flex">
          <a class="btn btn-outline-secondary" href="/logout">Logout</a>
        </div>
      </div>
    </nav>
    <div class="container mt-4">
      {% block content %}{% endblock %}
    </div>
  </body>
</html>
```

---

12) `web/templates/login.html`

```
{% extends 'base.html' %}
{% block content %}
<div class="row justify-content-center">
  <div class="col-md-6">
    <h3>Login</h3>
    <form method="post">
      <div class="mb-3">
        <label class="form-label">Username</label>
        <input class="form-control" name="username">
      </div>
      <div class="mb-3">
        <label class="form-label">Password</label>
        <input type="password" class="form-control" name="password">
      </div>
      <button class="btn btn-primary">Login</button>
    </form>
  </div>
</div>
{% endblock %}
```

---

13) `web/templates/dashboard.html`

```
{% extends 'base.html' %}
{% block content %}
<h2>Overview</h2>
<div class="row">
  <div class="col-md-6">
    <h4>Users</h4>
    <ul>
      {% for u in users %}
      <li>{{ u.id }} — lang: {{ u.lang if u.lang else 'ua' }}</li>
      {% endfor %}
    </ul>
  </div>
  <div class="col-md-6">
    <h4>Alerts</h4>
    <ul>
      {% for a in alerts %}
      <li>{{ a }}</li>
      {% endfor %}
    </ul>
  </div>
</div>
{% endblock %}
```

---

14) `core/__init__.py`

```python
# package init
from . import database
from .language import t, load_translations
```

---

15) `handlers/` (папка) — приклади

`handlers/README.md`

```
Тут можуть бути розбиті хендлери. У стартовому наборі ми використовуємо
bot.py як центральне місце.
```

---

16) `data/` — ця папка буде створена автоматично при першому запуску

---

17) Додаткові поради

- Якщо хочеш, я можу згенерувати повністю **новий** варіант з більш докладними хендлерами у `handlers/` (кожен файл окремо) — тоді код буде більший (~900-1100 рядків). Цей стартовий набір — компактний, але повнофункціональний: бот стартує, мова зберігається, Dashboard працює, є API-шляхи.

Кінець документу

```
```