Supptracker — Codex Task Pack

This single doc gives you PR-ready files, copy-paste Codex tasks, and exact commit/PR text. Drop these into the repo root and run the tasks in order.

---

0) Repo assumptions

- Python **3.11** (FastAPI backend).
- Node **20** (Vite/React frontend).
- Folders (adjust in prompts if yours differ):
- `backend/` — FastAPI app, data loaders
- `frontend/` — Vite app
- `tests/` — pytest tests

---

1) New file: `AGENTS.md`

```
# AGENTS.md

## Goals (Phase 1)
1. Fix synonym parsing (CSV uses commas; loader used `|`). Normalize and
dedupe.
2. Add real CI (Python + Node): lint/type/test/build.
3. Improve README quickstart with run/verify steps.
4. Optional: `docker-compose.yml` for local dev.

## Project Layout (expected)
- Backend (FastAPI) in `backend/`
- Frontend (Vite/React) in `frontend/`
- Tests in `tests/`

## Commands the agent should run
**Backend**
- Create venv: `python -m venv .venv && source .venv/bin/activate`
- Install: `pip install -r requirements.txt || pip install -e . || true`
- Tooling: `pip install pytest ruff mypy`
- Lint: `ruff check .`
- Typecheck: `mypy --strict || true`
- Test: `pytest -q`

**Frontend**
- `npm ci`
- Typecheck: `npx tsc --noEmit || true`
- Build: `npm run build`

## PR Rules
```

```
- Small, single-purpose PRs. One concern per PR.
- Pin action versions in CI; avoid leaking secrets.
- Include rationale and before/after in PR body.

## Acceptance Criteria
- Synonym parsing accepts `[|,;/]` as separators, trims, lowercases,
de-dupes.
- Unit test covers parsing of common herb aliases (e.g., St John’s wort →
hypericum).
- CI runs on PR and push; both backend and frontend jobs pass.
- README has 60-second quickstart + curl example for `/api/health`.
```

---

2) New file: `backend/synonyms.py`

Small, dependency-free utility the loader can call.

```
# backend/synonyms.py
from __future__ import annotations
import re
from typing import Iterable, List

SEP = re.compile(r"[|,;/]+")

def parse_synonyms(s: str | None) -> List[str]:
    """
    Split a raw synonyms string on common separators, trim/lowcase, and
dedupe.
    Returns sorted list for stable snapshots.
    """
    if not s:
        return []
    toks = [t.strip().lower() for t in SEP.split(s)]
    uniq = {t for t in toks if t}
    return sorted(uniq)


def normalize_names(names: Iterable[str]) -> List[str]:
    """Convenience: normalize an iterable of names/aliases to lookup keys."""
    out = []
    for n in names:
        if not n:
            continue
        out.append(n.strip().lower())
    # keep order but dedupe
    seen = set()
    norm = []
    for n in out:
        if n not in seen:
            seen.add(n)
            norm.append(n)
    return norm
```

3) New file: `tests/test_synonyms.py`

```
# tests/test_synonyms.py
from backend.synonyms import parse_synonyms, normalize_names


def test_parse_synonyms_handles_commas_pipes_semicolons():
    raw = "St John's wort, hypericum | Hypericum perforatum;  SJW "
    out = set(parse_synonyms(raw))
    assert {"st john's wort", "hypericum", "hypericum perforatum", "sjw"} <=
out


def test_normalize_names_dedupes_and_trims():
    names = [" Hypericum ", "hypericum", "SJW", "sjw", ""]
    out = normalize_names(names)
    assert out == ["hypericum", "sjw"]
```

4) (Optional) Patch hint: use the parser inside your loader

Agent instruction: Search for where ingredients/synonyms are parsed (likely in `backend/**/loader*.py` or similar) and replace any `.split('|')` logic with `parse_synonyms`. Example pattern below—**do not copy blindly**; integrate with your actual loader.

```
# backend/loader.py  (illustrative only)
from backend.synonyms import parse_synonyms, normalize_names

...
row_syn = parse_synonyms(row.get("synonyms"))
primary = (row.get("name") or "").strip().lower()
keys = normalize_names([primary, *row_syn])
# register all keys → primary id
for k in keys:
    index[k] = primary_id
```

5) New file: `.github/workflows/ci.yml`

```yaml
name: CI
on:
  push:
    branches: [ main ]
  pull_request:

jobs:
  backend:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-python@v5
        with:
          python-version: '3.11'
      - name: Cache pip
        uses: actions/cache@v4
        with:
          path: ~/.cache/pip
          key: ${{ runner.os }}-pip-${{ hashFiles('**/requirements*.txt') }}
      - name: Install deps
        run: |
          python -m venv .venv
          source .venv/bin/activate
          pip install -r requirements.txt || true
          pip install pytest ruff mypy
      - name: Lint
        run: |
          source .venv/bin/activate
          ruff check .
      - name: Typecheck (non-blocking)
        run: |
          source .venv/bin/activate
          mypy --strict || true
      - name: Test
        run: |
          source .venv/bin/activate
          pytest -q

  frontend:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: '20'
          cache: 'npm'
      - name: Install
        run: npm ci
      - name: Typecheck (non-blocking)
        run: npx tsc --noEmit || true
      - name: Build
        run: npm run build
```

---

6) New file: `docker-compose.yml` (optional but handy)

```
version: '3.9'
services:
  api:
    build: ./backend
    command: uvicorn app:app --host 0.0.0.0 --port 8000 --reload
    ports: ["8000:8000"]
    env_file: [backend/.env]
    volumes:
      - ./backend:/app
  web:
    build: ./frontend
    command: npm run dev -- --host
    ports: ["5173:5173"]
    volumes:
      - ./frontend:/app
    depends_on: [api]
```

---

7) README additions (Quickstart block)

Append this to your `README.md` under **Getting Started**.

~~~
### Quickstart (60 seconds)
**Backend**
```bash
python -m venv .venv && source .venv/bin/activate
pip install -r requirements.txt
uvicorn app:app --reload
~~~

Frontend

```
npm ci
npm run dev
```

Verify API

```
curl -s http://localhost:8000/api/health | jq
```

```
---

## 8) Codex (Agent) — Copy-paste tasks
Run these one by one. Each task creates a separate PR.

### Task 1 — Synonym bugfix + tests
**Instruction to Codex:**
```

Goal: Fix synonym parsing and add tests. Repo layout: backend(FastAPI), frontend(Vite), tests(pytest).

Steps: 1) Add `backend/synonyms.py` exactly as given in our spec (see doc). 2) Replace any `.split('|')` or ad-hoc synonym parsing in the data loader with calls to `parse_synonyms` and `normalize_names`. 3) Add `tests/test_synonyms.py` from the spec. 4) Ensure `pytest -q` passes locally. 5) Open a PR titled: "fix(loader): robust synonym parsing + unit tests" with a summary of the change and before/after example.

```
**Commit message suggestion:**
```

fix(loader): robust synonym parsing + unit tests

- accept [|,;/] separators; trim/lowcase; dedupe
- add `backend/synonyms.py` utility module
- replace split logic in loader to use shared parser
- add tests for parsing + normalization

  ```
  ---

  ### Task 2 — Real CI for backend & frontend
  **Instruction to Codex:**
  ```

  Goal: Add real CI (GitHub Actions) for backend and frontend.

Steps: 1) Create `.github/workflows/ci.yml` as in our spec. 2) Ensure it runs on push and PR; separate jobs for Python and Node. 3) Keep mypy/tsc non-blocking (do not fail the build on type warnings for now). 4) Open a PR titled: "ci: add backend+frontend pipeline (lint, type, test, build)".

```
**Commit message suggestion:**
```

ci: add backend+frontend pipeline (lint, type, test, build)

- backend job: ruff, mypy (non-blocking), pytest
- frontend job: tsc (non-blocking), build

```
---

### Task 3 — README quickstart
**Instruction to Codex:**
```

Goal: Improve README with a minimal quickstart and healthcheck.

Steps: 1) Append the Quickstart block to README. 2) Ensure commands match repo scripts (uvicorn app path, npm scripts). 3) Open a PR titled: "docs: add 60-second quickstart and API healthcheck".

```
**Commit message suggestion:**
```

docs: add 60-second quickstart and API healthcheck

```
---

### Task 4 (Optional) — `docker-compose.yml`
**Instruction to Codex:**
```

Goal: Add docker-compose for local dev.

Steps: 1) Create `docker-compose.yml` from the spec, adjusting app entrypoints if needed. 2) Add a README section: `docker-compose up` instructions. 3) Open a PR titled: "dev: docker-compose for api+web".

```
**Commit message suggestion:**
```

dev: add docker-compose for api+web local dev

~~~
---

## 9) Local run cheat sheet (no agent required)
```bash
# Backend
python -m venv .venv && source .venv/bin/activate
pip install -r requirements.txt
pytest -q
uvicorn app:app --reload

# Frontend
npm ci
npm run dev

# Health check
curl -s http://localhost:8000/api/health | jq
~~~

---

10) Post-merge checklist

- Tag `v0.1.0` after Tasks 1–3 are merged.
- Create GitHub Release notes summarizing fixes and CI.
- Open follow-up issues:
- Integrate synonym mapping into `/api/interaction` end-to-end test.
- Add `rules.yaml` schema validation.
- Seed data integrity check on startup.