ок, тримай все “від А до Я” прямо в чаті — просто скопіюй це в свій документ. Я зробив як **специфікацію + короткі готові шматки коду** (ES5-сумісні).

**Contract AI — Word Taskpane: логіка і план змін (оновлено 2025-08-10)**

**1) Ціль та очікуваний результат**

* Користувач задає **Backend URL** → **Save** зберігає у localStorage.
* **Test backend** робить GET /health і показує результат.
* В полі **Original clause** користувач вставляє текст або натискає **Use selection →** (читає виділення з Word).
* **Analyze** → POST /api/analyze → показати висновки.
* **Get AI Draft** → POST /api/gpt/draft → показати чернетку.
* **Insert result into Word** вставляє результат у поточне виділення документа.

**2) Архітектура та файли**

* **manifest.xml** — реєструє надбудову, вказує шлях до панелі.
* **taskpane.html** — весь UI + XHR мережева логіка + (опційно) Office.js.
* **Frontend HTTPS (порт 3000)** — віддає статичні файли.
* **Backend HTTPS (порт 9000)** — GET /health, POST /api/analyze, POST /api/gpt/draft.

**3) Послідовність завантаження (boot)**

1. DOMContentLoaded → init() (працює **без Office.js**).
2. Кнопки **Save**/**Test backend** активні завжди.
3. Паралельно: якщо доступний Office.js → Office.onReady() (таймаут 5с).
   * Якщо ОК → показуємо блок “Office API is ready” і вмикаємо дії з документом.
   * Якщо нема/таймаут → працюємо в **standalone** (без Word API).
4. window.onerror → усі помилки з’являються у **Status** та **Console**.
5. “Heartbeat”: таймер у document.title (видно, що JS живий).

**4) UI/UX (елементи і стани)**

* **Backend URL + Save + Test backend** (з localStorage).
* **Original clause** (textarea) + **Use selection →** (коли Office готовий).
* Кнопки **Analyze** (→ /api/analyze) і **Get AI Draft** (→ /api/gpt/draft).
* **Result** (textarea) + **Insert result into Word**.
* **Status** (коротке повідомлення) + **Console** (деталі).

UX-удосконалення (після базової перевірки): дизейбл кнопок під час запиту, індикатор прогресу.

**5) Мережа (тільки XHR, без fetch)**

* **Health**: GET {base}/health (таймаут 5–6с). Очікуємо 200 {"status":"ok"}.
* **Analyze**: POST {base}/api/analyze (таймаут 20–25с).
* **Draft**: POST {base}/api/gpt/draft (таймаут 30–60с).
* Тип тіла: application/json.
* **TLS**: dev-сертифікат має бути **довірений у Windows** (і для 3000, і для 9000).
* **CORS** на бекенді:
* Access-Control-Allow-Origin: https://localhost:3000
* Access-Control-Allow-Methods: GET, POST, OPTIONS
* Access-Control-Allow-Headers: Content-Type, Authorization

і відповідь 200 на OPTIONS.

**6) Інтеграція з Word (опційно, якщо є Office.js)**

* **Use selection →** читає document.getSelection().text у textarea.
* **Insert result** робить insertText(result, "Replace").
* Ці дії **з’являються тільки** після успішного Office.onReady.

**7) Кеш і маніфест**

* У маніфесті використовуємо **cache-busting** і бамп версії:
* <Version>1.0.0.3</Version>
* <DefaultSettings>
* <SourceLocation DefaultValue="https://localhost:3000/taskpane.html?v=build-2025-08-10-2013"/>
* </DefaultSettings>
* Після будь-яких змін HTML/JS змінюй v=....
* Для “чистого” рестарту: закрий Word → видали %LOCALAPPDATA%\Microsoft\Office\16.0\Wef\\* → знову sideload.

**8) Повідомлення про помилки (що показуємо)**

* ✅ Health OK (200): ... або ⚠️ HTTP 4xx/5xx (мережа є, API відповідає, але з помилкою).
* ❌ Request failed: timeout/network error (нема з’єднання/серти/фаєрвол).
* Для Word-операцій: ⚠️ Word API not available (якщо Office.js не піднявся).

**9) Acceptance tests (чек-лист)**

1. Панель вантажиться; видно **Build**, **Save**, **Test backend**.
2. **Test backend** на твоєму /health дає ✅ 200.
3. Після завантаження Office.js видно “Office API is ready”.
4. **Use selection →** переносить виділення у textarea.
5. **Analyze** → отримуємо відповідь (або осмислену помилку).
6. **Get AI Draft** → отримуємо чернетку.
7. **Insert result** вставляє результат у документ.
8. Перезапуск Word — **Backend URL** зберігається в localStorage.

**10) Що змінили порівняно з початковою панеллю**

* Додали **standalone-режим** (панель живе без Office.js).
* Уся мережа переведена на **XHR (ES5)**.
* Додано Health і чіткі статуси.
* Операції з Word вмикаються **після** Office.onReady.
* Впроваджений **cache-busting** у маніфесті та практика бампу версій.

**Готові фрагменти коду (вставляй як є)**

**1) Маніфест (уривок)**

<?xml version="1.0" encoding="UTF-8"?>

<OfficeApp xmlns="http://schemas.microsoft.com/office/appforoffice/1.1"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:type="TaskPaneApp">

<Id>3a7f2ae0-2d33-4d6e-8c0b-3d0a3a8dbe9f</Id>

<Version>1.0.0.3</Version>

<ProviderName>Contract AI</ProviderName>

<DefaultLocale>en-US</DefaultLocale>

<DisplayName DefaultValue="Contract AI — Draft Assistant"/>

<Description DefaultValue="Analyze a clause and get an AI draft."/>

<AppDomains>

<AppDomain>https://localhost:3000</AppDomain>

<AppDomain>https://localhost:9000</AppDomain>

</AppDomains>

<Hosts><Host Name="Document"/></Hosts>

<DefaultSettings>

<SourceLocation DefaultValue="https://localhost:3000/taskpane.html?v=build-2025-08-10-2013"/>

</DefaultSettings>

<Permissions>ReadWriteDocument</Permissions>

</OfficeApp>

**2) Універсальні XHR-функції (ES5)**

<script>

function httpGet(url, timeoutMs){

return new Promise(function(resolve, reject){

try{

var xhr = new XMLHttpRequest();

xhr.open("GET", url, true);

xhr.timeout = timeoutMs || 6000;

xhr.onreadystatechange = function(){

if(xhr.readyState === 4){

resolve({ ok: (xhr.status>=200 && xhr.status<300), status: xhr.status, text: xhr.responseText });

}

};

xhr.ontimeout = function(){ reject(new Error("timeout")); };

xhr.onerror = function(){ reject(new Error("network error")); };

xhr.send();

}catch(e){ reject(e); }

});

}

function postJson(url, data, timeoutMs){

return new Promise(function(resolve, reject){

try{

var xhr = new XMLHttpRequest();

xhr.open("POST", url, true);

xhr.setRequestHeader("Content-Type","application/json");

xhr.timeout = timeoutMs || 30000;

xhr.onreadystatechange = function(){

if(xhr.readyState === 4){

var text = xhr.responseText || "";

var json = null;

try{ json = text ? JSON.parse(text) : null; }catch(e){}

resolve({ ok: (xhr.status>=200 && xhr.status<300), status: xhr.status, text: text, json: json });

}

};

xhr.ontimeout = function(){ reject(new Error("timeout")); };

xhr.onerror = function(){ reject(new Error("network error")); };

xhr.send(JSON.stringify(data||{}));

}catch(e){ reject(e); }

});

}

</script>

**3) Save / Test backend**

<script>

function baseUrl(){ return (document.getElementById("backendUrl").value||"").trim().replace(/\/$/,""); }

function setStatus(msg){ document.getElementById("status").textContent = msg; }

function saveBackend(){

var v = baseUrl();

if(!v){ setStatus("⚠️ Enter backend URL first."); return; }

try{ localStorage.setItem("contract\_ai\_backend", v); setStatus("✅ Saved: " + v); }

catch(e){ setStatus("❌ localStorage error"); }

}

function testBackend(){

var v = baseUrl();

if(!v){ setStatus("⚠️ Enter URL, e.g. https://localhost:9000"); return; }

setStatus("⏳ Testing " + v + " /health …");

httpGet(v + "/health", 6000).then(function(res){

setStatus(res.ok ? ("✅ Health OK ("+res.status+"): "+(res.text||"(empty)"))

: ("⚠️ HTTP "+res.status+" — "+(res.text||"(empty)")));

}).catch(function(err){ setStatus("❌ Request failed: " + (err.message||err)); });

}

</script>

**4) Analyze / Draft (POST)**

<script>

function analyze(){

var v = baseUrl(); var text = (document.getElementById("clauseBox").value||"").trim();

if(!v){ setStatus("⚠️ Set backend URL first"); return; }

if(!text){ setStatus("⚠️ Enter or load text first"); return; }

setStatus("⏳ Analyzing …");

postJson(v + "/api/analyze", { text: text }, 25000).then(function(res){

document.getElementById("resultBox").value = res.text || (res.json?JSON.stringify(res.json,null,2):"");

setStatus(res.ok ? ("✅ Analyze OK ("+res.status+")") : ("⚠️ Analyze HTTP "+res.status));

}).catch(function(err){ setStatus("❌ Analyze failed: " + (err.message||err)); });

}

function draft(){

var v = baseUrl(); var text = (document.getElementById("clauseBox").value||"").trim();

if(!v){ setStatus("⚠️ Set backend URL first"); return; }

if(!text){ setStatus("⚠️ Enter or load text first"); return; }

setStatus("⏳ Drafting …");

postJson(v + "/api/gpt/draft", { text: text }, 60000).then(function(res){

document.getElementById("resultBox").value = res.text || (res.json?JSON.stringify(res.json,null,2):"");

setStatus(res.ok ? ("✅ Draft OK ("+res.status+")") : ("⚠️ Draft HTTP "+res.status));

}).catch(function(err){ setStatus("❌ Draft failed: " + (err.message||err)); });

}

</script>

**5) Office.js (опційно): onReady + Use selection + Insert**

<script src="https://appsforoffice.microsoft.com/lib/1/hosted/office.js" defer></script>

<script>

function tryInitOffice(){

if(!window.Office || !Office.onReady){ console.log("Office.js not loaded"); return; }

var armed = true;

setTimeout(function(){ if(armed){ console.log("Office.onReady timeout"); }}, 5000);

Office.onReady(function(){

armed = false;

document.getElementById("officeBlock").style.display = "";

});

}

function useSelectionToBox(){

if(!window.Word || !Word.run){ setStatus("⚠️ Word API not available"); return; }

Word.run(function(ctx){

var sel = ctx.document.getSelection(); sel.load("text");

return ctx.sync().then(function(){

document.getElementById("clauseBox").value = sel.text || "";

setStatus("✅ Selection copied");

});

}).catch(function(e){ setStatus("❌ Read selection failed: " + (e.message||e)); });

}

function insertResult(){

var text = (document.getElementById("resultBox").value||"").trim();

if(!text){ setStatus("⚠️ Nothing to insert"); return; }

if(!window.Word || !Word.run){ setStatus("⚠️ Word API not available"); return; }

Word.run(function(ctx){

var sel = ctx.document.getSelection();

sel.insertText(text, "Replace");

return ctx.sync();

}).then(function(){ setStatus("✅ Inserted into Word."); })

.catch(function(e){ setStatus("❌ Insert failed: " + (e.message||e)); });

}

document.addEventListener("DOMContentLoaded", function(){

tryInitOffice();

});

</script>

**11) Наступні кроки**

* Додати **Bearer-авторизацію** (поле для токена → зберігати в sessionStorage, додавати заголовок Authorization).
* Додати дизейбл кнопок під час запиту + простий прогрес-бар.
* За потреби — “Network debug” (час відповіді, заголовки).

якщо хочеш, я з цього зроблю ще й **єдиний taskpane.html** (повна панель) — скажи, чи включати авторизацію відразу і який точний формат тіла/відповіді у твоїх /api/analyze та /api/gpt/draft.

Рігення що спрацювало коли біли мертві кнопки внизу. Є! Тепер видно справжню причину: **/health працює (GET 200), а Analyze “висить”**. Це типовий кейс **CORS-префлайту**:

* Панель грузиться з [**https://localhost:3000**](https://localhost:3000/).
* Analyze робить **POST** на [**https://localhost:9000**](https://localhost:9000/) з JSON → браузер спершу шле **OPTIONS (preflight)**.
* Якщо на бекенді **не ввімкнено CORS**, браузер блокує POST, і в панелі нескінченне “Analyzing…”.

**Рішення (1 хв): увімкнути CORS у FastAPI**

**Мінімальна правка в app.py (додай один блок — і все оживе)**

# app.py — ДОДАТИ НА ПОЧАТКУ, після створення FastAPI() або одразу після імпортів

from fastapi import FastAPI

from fastapi.middleware.cors import CORSMiddleware

app = FastAPI()

# Додай обидва домени (localhost і 127.0.0.1) з портом 3000

ALLOWED\_ORIGINS = [

"https://localhost:3000",

"https://127.0.0.1:3000",

]

app.add\_middleware(

CORSMiddleware,

allow\_origins=ALLOWED\_ORIGINS,

allow\_credentials=True,

allow\_methods=["GET", "POST", "OPTIONS"],

allow\_headers=["Content-Type", "Authorization"],

)

Важливо: **залиш все інше як є** (маршрути /health, /api/analyze, /api/gpt/draft не чіпаємо). Ми лише додаємо CORS-middleware.

**Навіщо саме так**

* **GET /health** не потребує префлайту → тому працював.
* **POST /api/analyze** з Content-Type: application/json викликає **OPTIONS** → без CORSMiddleware відповідь не приходить, браузер блокує запит.

**Що зробити по кроках**

1. Відкрий app.py, додай блок CORSMiddleware (як вище).
2. Зупини бекенд (вікно Uvicorn) і запусти знову твоїм **RUN.bat** (він усе перезапустить).
3. У панелі:
   * натисни **Test** → має бути ✅ Backend connected (200).
   * натисни **Analyze** → статус зміниться на ⏳ Analyzing…, далі ✅ Analyze OK. — і з’являться бейджі/дані.
   * **Get AI Draft** → ✅ Draft OK.

Якщо після цього все ще “висить” — скажи, що показує рядок **Status** під час Analyze (я там виводжу точний текст помилки). Але за симптомами це саме CORS.

**STATE LOG**

* **Причина:** браузерний **CORS** блокує POST /api/analyze (префлайт OPTIONS), тому в панелі “Analyzing…” без відповіді, тоді як GET /health працює.
* **Фікс:** додати CORSMiddleware у app.py з allow\_origins=["https://localhost:3000","https://127.0.0.1:3000"], allow\_methods=["GET","POST","OPTIONS"], allow\_headers=["Content-Type","Authorization"].
* **TODO:** вставити блок, перезапустити бекенд через RUN.bat, натиснути **Analyze**.
* **Ризики:** якщо фронт іноді відкривається з http:// (без https) — додати і http://localhost:3000 у allow\_origins.