# Initial Report – HubSpot Public App ↔ Titanx

## 1. Context

The goal is to build a **public app** for the HubSpot App Marketplace that connects HubSpot with Titanx.  
The objective is for users to install it directly from the Marketplace, authorize it via OAuth 2.0, and manage data synchronization (e.g., contacts, deals, properties, business-specific operations).

The design must be **multi-tenant**, allowing multiple HubSpot portals to connect independently.

## 2. Account types and licenses required

### To create the Public App

* **HubSpot Developer Account (free):**  
  Required to register the application, define scopes, test OAuth, and prepare for Marketplace publishing.

### To use the Public App

* **Any HubSpot portal (Free, Starter, Pro, or Enterprise):**
  + An account with **Super Admin** permissions can install the app.
  + The app can work in Free accounts, but **features depend on the license**:
    - **Free/Starter:** only basic access to contacts and standard properties.
    - **Professional/Enterprise:** advanced access (e.g., workflows, custom objects, advanced reporting).

⚠️ If the app integrates with **custom objects** or advanced APIs, the portal must be **Professional or Enterprise**.

## 3. UI and user experience

* With **HubSpot UI Extensions**, we can render pages inside **Settings** and in specific records (e.g., contact or deal sidebar).
* **Limitation:** if a more complex UI is needed (advanced dashboards, visual reports, heavy data handling), a **separate external app** (e.g., React/Next.js webapp) will be required to connect to the backend and display the data outside HubSpot.

## 4. External backend required

For a Public App to function properly, an **external backend** is required, with the following characteristics:

* **AWS (EC2, Lambda, or ECS)** for scalability.
* **Database:** (multi-tenant) to store access/refresh tokens per portal.
* **Authentication:** full implementation of the **HubSpot OAuth 2.0 flow**.
* **Mandatory requirements:**
  + Active **SSL** (HTTPS).
  + **Custom domain** (e.g., api.mydomain.com).
  + **High availability** (99.9% uptime or better).
  + Logging and monitoring (e.g., CloudWatch or ELK stack).

The backend will handle: - Storing and refreshing tokens per connected portal.  
- Exposing endpoints for HubSpot UI consumption (via hubspot.fetch).  
- Connecting with Titanx for bidirectional synchronization.

## 5. Best practices for Marketplace publishing

* **Security:**
  + Store tokens encrypted in the database.
  + Strict refresh token handling.
  + HTTPS on all requests.
* **UX:**
  + 3-step installation (Install → OAuth → Settings page).
  + Clear “Reconnect” button if a token expires.
* **HubSpot Marketplace requirements:**
  + Public privacy policy.
  + Support URL and documentation.
  + Demo video (installation & usage).
  + Validation and approval by HubSpot’s team.

## 6. Pending definition

The API itself isn’t incomplete, but the **functional flow is undefined**:  
- Which HubSpot entities should sync with Titanx (contacts, deals, tickets, custom objects)?  
- What is the data direction (read-only, write-only, or bidirectional)?  
- With what frequency and under what business rules?  
- How should data conflicts be handled?

## 7. Critical questions for Titanx

| Question | Why it’s critical / what it affects | Expected information |
| --- | --- | --- |
| **Which endpoints will I use for scoring / enrich / sync (methods, routes, payloads, responses)?** | Without this, backend design and data mapping are impossible. | Example JSON request/response, errors, limits. |
| **What data (fields) can/should I sync between Titanx ↔ HubSpot?** | Required to define HubSpot properties, mappings, and OAuth scopes. | List of fields with types, required/optional, length. |
| **What events trigger notifications towards you (webhooks)?** | Defines when to receive push data instead of polling. | Webhook URLs, available events, expected payloads. |
| **Are there usage limits / API rate limits?** | Needed to design throttling / backoff / queuing. | Per minute/hour quota, quota headers, reset behavior. |
| **Are there sandbox / test / production environments?** | Avoids breaking production during testing. | Domains, dummy credentials, separate keys. |
| **What business logic / rules exist for scoring / enrichment?** | Needed to expose correct logic in UI and backend. | Formulas, when to re-evaluate, when to invalidate. |
| **What identifier is used to link entities with HubSpot (email, uuid, external\_id)?** | Required for matching HubSpot data with Titanx. | Field like “titanx\_id” or matching rules. |
| **What sync frequency is allowed / expected?** | Needed to size processes (batch vs. real-time). | Example: every 5 min, event-triggered, nightly batch. |
| **What behavior is expected on failure / data inconsistency?** | Defines error tolerance and fault handling. | Retries, dead letters, logs, alerts. |

## 8. Development timeline

The complete project is estimated at **5–6 months**, broken down into phases:

1. **Month 1–2:**
   * App flow design.
   * Backend setup in AWS with database(MongoDB, PostgreSQL, MySQL).
   * Multi-tenant OAuth implementation.
2. **Month 3–4:**
   * Backend endpoint development.
   * HubSpot UI (Settings + contextual panels).
   * Titanx integration.
3. **Month 5–6:**
   * QA, performance & security testing.
   * Documentation and demo video prep.
   * Submission and approval in Marketplace.

## 9. Next steps

1. Define the **functional flow** of the app with the product team.
2. Confirm the **minimum HubSpot license requirement** for customers.
3. Deploy initial backend in AWS with SSL domain.
4. Build **MVP authentication + token storage**.