**Expectations for the Technical Data Documentation:**

**1. Blueprint / Architecture Overview**

* A high-level **system architecture diagram**
  + Show how data flows: from pharmacies → through wholesalers → to your platform.
  + Highlight any third-party services, APIs, or data processors involved.
* Mention how **data is stored**, including:
  + Is it centralized or decentralized?
  + What kind of databases are used?
  + How is data anonymized or pseudonymized, if at all?

**2. Data Requirements**

* A **clear list of data points** needed from each wholesaler:
  + Transaction data (e.g., product, quantity, timestamp)
  + Pricing?
  + Inventory levels?
  + Pharmacy ID?
* Clarify which data is **essential vs. optional**.

**3. Integration Expectations**

* **How your system receives data**:
  + API?
  + SFTP?
  + Manual upload?
* Frequency of data sharing (real-time, daily batch, etc.)
* Level of automation required from the distributor’s end.

**4. Security & Compliance**

* What measures are in place to **protect sensitive data**?
  + Encryption (in transit and at rest)
  + Access controls
  + Compliance with GDPR or other relevant regulations
* Who has access to the data, and how is it audited?

**List known unknowns**:

* + What do you *not* know yet about distributor data?
  + What assumptions are you making?
  + This shows you’re thinking responsibly and collaboratively.

**Create a sample data exchange spec**:

* + What would a "perfect" data file or API call look like?
  + Include field names, data types, and a mock example.

**Purpose**

A short paragraph (2–3 sentences) describing the goal of the system and why this data exchange is important.

Answer:

Tech Stacks: React   
Deployment: Netlify

**Cloud Service Providers:**

Shared Responsibility Model. – underlining infrastructure is compliant already.

We inherit these in our system.

For GDPR -> we need to follow some configuration steps in AWS (like a checklist) [Eg: Encryptions – in diff layers (REST)]

Show -> Service – Configurations – Our algorithm – Our external APIs interacting with the data

Permissions/ Compliances -> Who? Why ? How? (Admins of the Platform – Developers ) -> Authorisation id mention (just like a hash id- primary key) – Access Layer

Second Layers of the Application:

Recommendation: Cheapest & cleanest -> **Serverless services in AWS** (Arch 1) – Customers loves this feedback – scale quick.

<https://aws.amazon.com/getting-started/hands-on/build-serverless-web-app-lambda-amplify-bedrock-cognito-gen-ai/>

<https://aws.amazon.com/lambda/web-apps/>

or Monolith on EC2 (Arch 2)