1. Models and Serialization (JSON/XML)

The core of our API design is to expose our Django models (`Store`, `Product`, and `Review`) in a machine-readable format. We will use Django REST Framework's `ModelSerializer` to achieve this.

The primary format for our resources will be **JSON (JavaScript Object Notation)**, as it is lightweight, human-readable, and widely supported by modern web applications. We have also configured the API to support **XML (Extensible Markup Language)** for vendors who may prefer a different format.

- **Store Model**: The `StoreSerializer` will convert `Store` model instances into a JSON object containing the `vendor`ID, `name`, and `description`.
- **Product Model**: The `ProductSerializer` will serialize `Product` instances, exposing the `store` ID, `name`, `description`, and `price`.
- **Review Model**: The `ReviewSerializer` will convert `Review` objects, representing the `product` ID, `user` ID, review `text`, and `is_verified` status.

---

2. API Endpoints

Our API endpoints are designed to be clear and follow **RESTful best practices**, using specific URLs to represent resources and standard HTTP methods to perform actions on them.

- **For Creating Resources (Authentication Required):**
  - `POST /api/stores/create/`: Allows a vendor to create a new store by sending a JSON payload.
  - `POST /api/products/add/`: Allows a vendor to add a new product to an existing store.
- **For Retrieving Resources (Public Access):**
  - `GET /api/stores/all/`: Retrieves a list of all stores from every vendor.
  - `GET /api/products/all/`: Retrieves a list of all products from every store.
- **For Retrieving Resources (Authentication Required):**
  - `GET /api/reviews/vendor/`: Allows a vendor to retrieve all reviews for their own products.
  - `GET /api/stores/vendor/`: Allows a vendor to retrieve a list of only their own stores.
  - `GET /api/products/vendor/xml/`: This unique endpoint allows a vendor to retrieve their products in **XML format**, demonstrating support for multiple serialization formats.

Each endpoint is designed to be self-explanatory and to handle the specific permissions required for its action.