Problem Statement

Design a backend service for the following business use case -

At OLX we want to build an Inventory management system. Which will deal with the creation of inventory, updation of inventory statuses and locations. An inventory will have an primary status i.e. CREATED, PROCURED and SOLD and also can have multiple secondary statuses *(more details follows below)*.

Inventory should have a SKU (Stock Keep Unit) Identifier, a type, a primary status, a primary location, a set of attributes like for a *car type* inventory we can have - *VIN (Vehicle Identification Number) & MMTY (make, model, trim & year)*, and some pricing details (ie. cost & selling price), it can additionally have some metadata like created at time, last updated at time, which user created inventory and last updated by which user.

Task -
DB - Create a DB schema for the following problem stated above.

REST API -

1) We need to create CRUD APIs for Inventory.
    a) Create inventory API.
    b) Get Inventory via SKU API.
    c) Get All Inventories API with pagination (10 Inventory per request).
    d) Update inventory API ie, status, attributes & pricing.
2) Create and update secondary status API.
3) Create a dashboard to view the inventory data using redash.(optional)

Note -

- We also need to implement proper error handling for all APIs, in case of error we should get error response in JSON Object format with errorCode & errMessage attributes, with correct status.

Example - 404 Error

```
{
    errorCode: “NotFound”
    errorMessage: “Inventory not found”
}
```

Secondary statuses

- The secondary statuses represent the statues from other systems
    - For example
        - Warehouse - IN_REPAIR
        - TRANSIT - COMPLETE
    - It consists of a system name and the status value.
    - We can update the status for a particular system through API.

Additional task -

- Create user login & registration APIs with Basic User schema in DB with email and password
- For login use JWT authentication.
- Add authentication in the Inventories APIs and those APIs should be authenticated with the JWT token created at login.

Things to keep in mind -

- Make small commits and keep pushing on the remote repository.
- Code should have 100% test coverage.
- Keep clean code principles in mind.
- Use IDE shortcuts.
- Use correct REST principles ie URLs, HTTP Verb & Response Code.