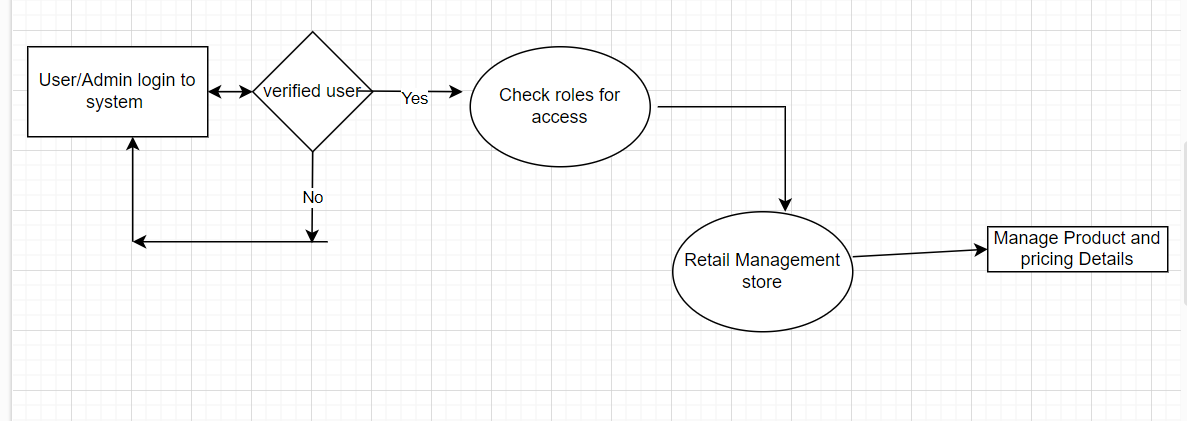
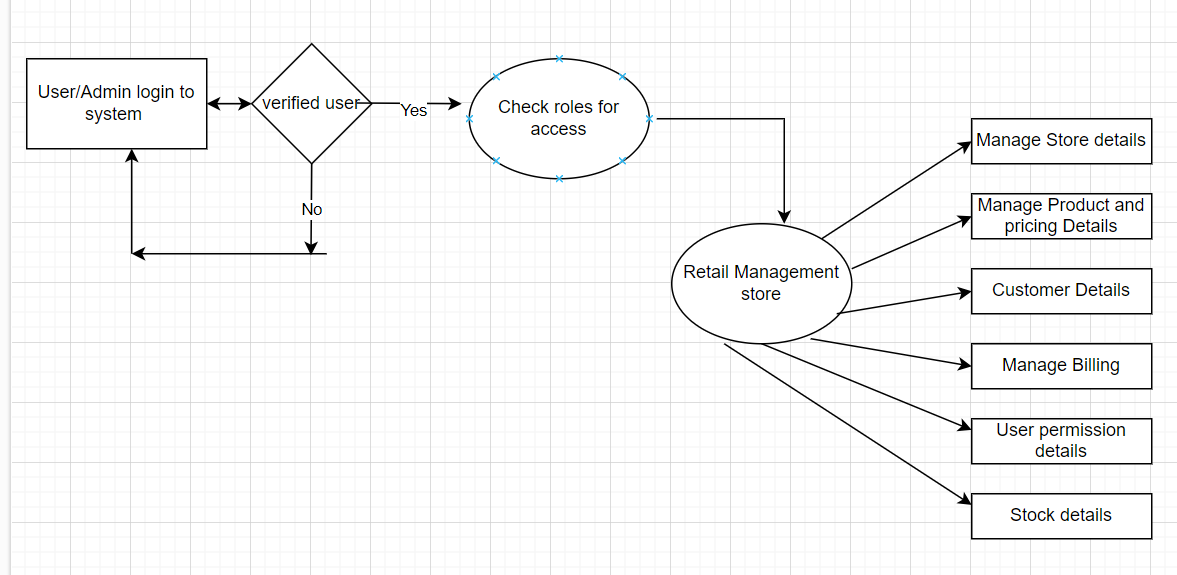
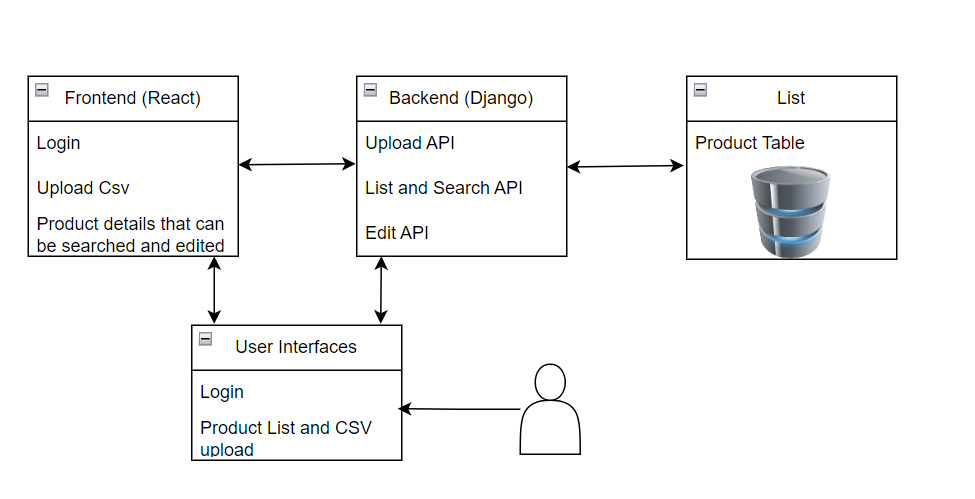
**Context Diagram:**



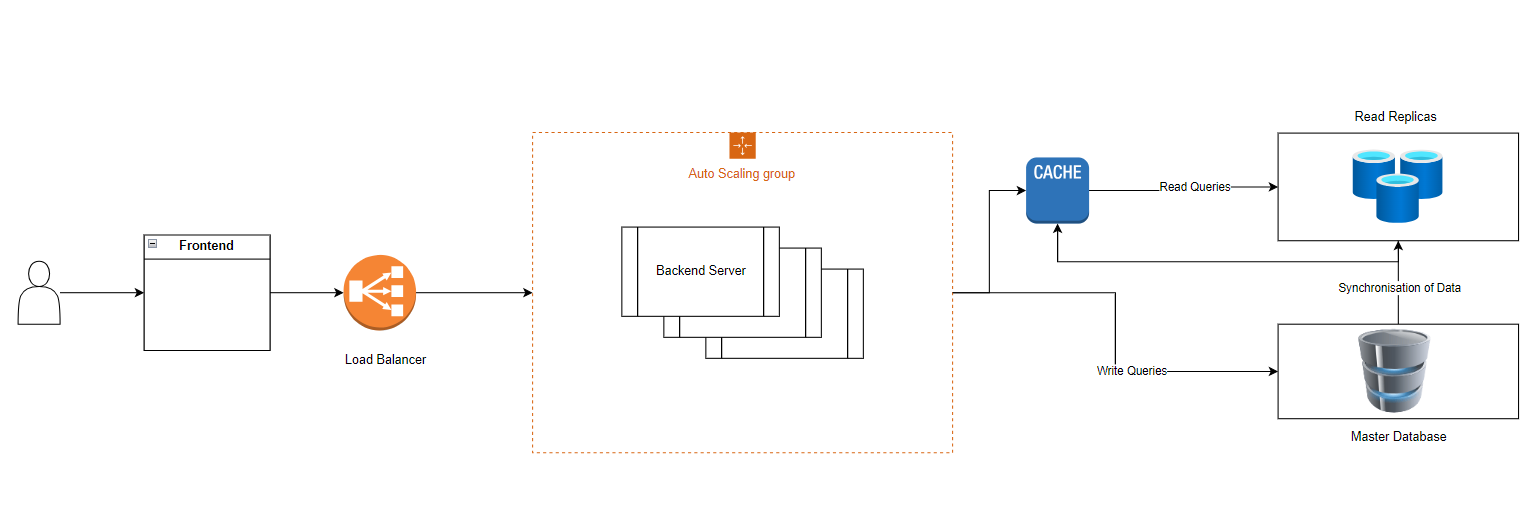
Considering Non functional requirements



**Architecture :**



**Non-functional requirements design Discussion**



**Description:**

Frontend (React):

* Login Form
* Upload From: Allows user to upload CSV file containing STORE\_ID,SKU,PRODUCT NAME,PRICE,DATE
* Product list with search and edit capability

Load Balancer:

* Distributing incoming requests from frontend to multiple backend servers to ensure even load distributed, high availability and scalability.

Backend (Django):

* Upload API: Receives csv file uploads.
* Search API: List the Product data and search the pricing or search using other criteria.
* Edit API: Edit the product data.

Database:

* Primary db: Handles write operations to product table
* Read Replica:

1. Handles read heavy queries to reduce load on primary dB and improve performance for read operations.
2. Direct all the read queries to replicas(slave nodes) of primary db.
3. Propagate the changes from primary db to replicas as and when they happen.

**Assumptions:**

1. Considering pricing feeds are uploaded once a day and queried lot of times it is read heavy, so adding replicas of primary db.
2. Uploaded file is CSV and within acceptable size limits.
3. Creating Index for columns to speed up data retrieval.

**Source of Implementation:**

1.React : v18

2.Material UI: v5.16

2.Django: v5

3.Django rest framework

4.Postgres SQL