Food Warehouse Management system

Technologies used:

React JS (Frontend), Node JS & Express JS (Backend Connections), MongoDB (Backend)

Source Code: https://github.com/Swapnilkulkarni-github/Food-Wearhouse.git

**How to run:** (Pre-requisites ‘Node’ must be installed in system)

1. Extract given Zip
2. Inside folder “Food-Warehouse” , Frontend and Backend are two folders open them in any IDE (used VS Code)
3. Frontend : inside Frontend folder run command ‘npm install’ 🡪 ‘npm start’

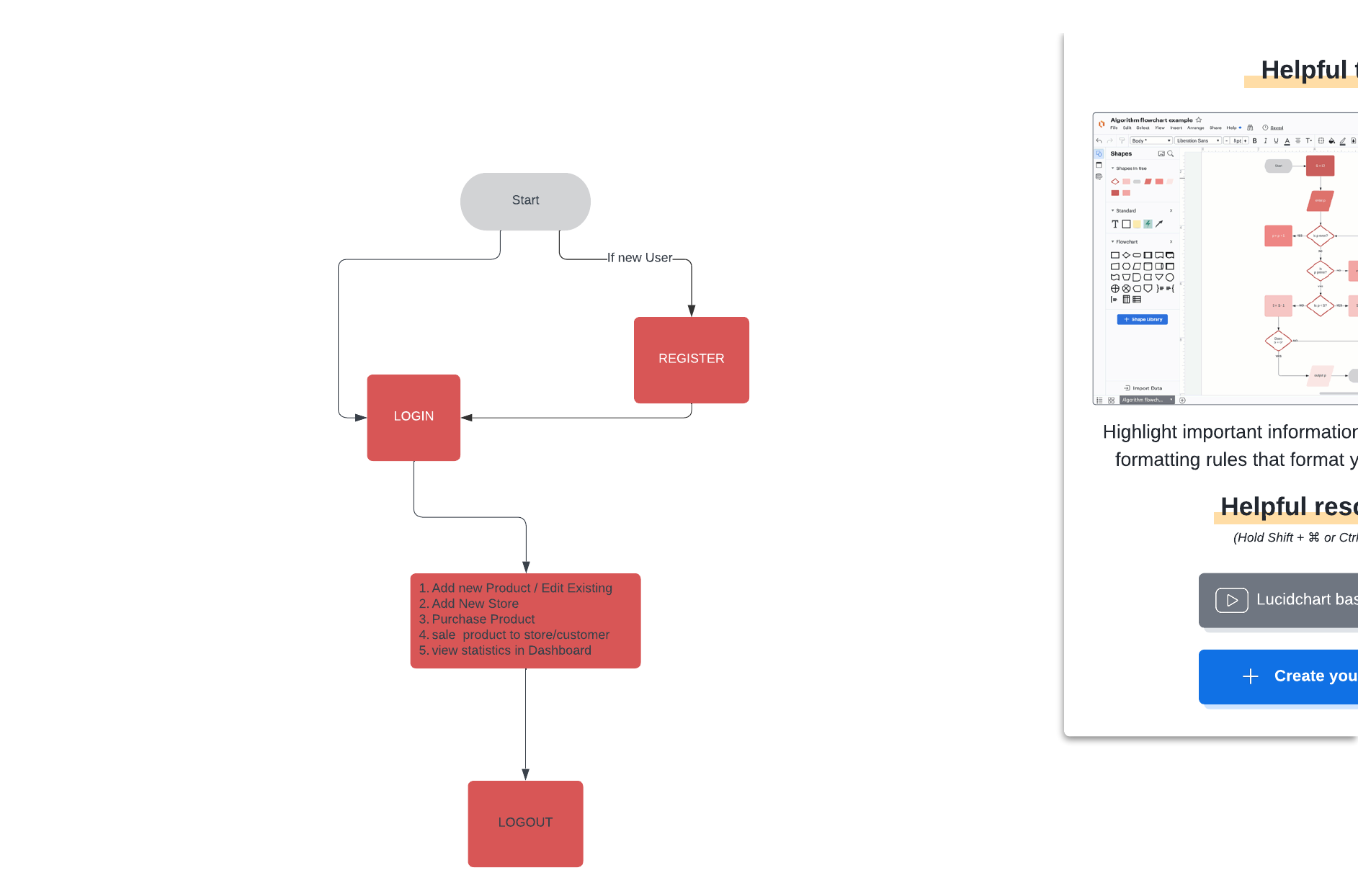
(Front end will start on localhost/3000)

1. Backend :
   1. Need to setup Mongo DB create MongoDB atlas account and sign in
   2. After sign in and you’ll see a default cluster click Connect 🡪 Drivers 🡪Copy Connection URL 🡪 paste in backend/models/index.js file

URL needs editing: add data username , password and filename at respective places mongodb+srv://USERNAME:PASSWORD@cluster0.gmjmgio.mongodb.net/FILENAME?retryWrites=true&w=majority

* 1. After all configuration , Inside Backend folder run command ‘npm install’ 🡪 ‘nodemon server.js’ (backend will be start on localhost/4000)

**System Flow chart:**

****

**How to Use**: Food-Warehouse Management System

* Go to URL localhost/3000, as a 1st time visitor direct to Registration page, After Registering with appropriate data Login to System.

Step 1: Inside inventory tab, create products by clicking on Add Product button with Proper data

Step 2: Inside Manage Store tab, Add Store to whom the warehouse will send products.

Step 3: After this setup now you can add sales (warehouse sending products to store) and add purchase (warehouse ordering material)

**Functionalities:**

* User can Login and Register
* User can add, Edit and Delete Product
* User can Track Sold products to stores and customer
* Managing Inventory purchase of product
* Can see data statically in dashboard

**API Information:**

* All the API code will be found in backend folder of given code
* Used MVC (Model view controller) Architecture to design and break-up code logics into separate components
* In Model Folder ,we have all definitions and information about our documents(data tables) our main Tables are USER , PRODUCT , PURCHASE , SALES , STORE
* In the routes folder, we have different routes according to model like in PRODUCT route we have all relates routes defined like add , edit , delete , view
* In Controller folder , there are actual functions written to manipulate data for each route and model

**Testing**:

* To test system insufficient and wrong data submission would work like not completely entering required fields.
* On the clicks page being rendered properly or not can also be a scenario.