DASS Assignment 2

Bulk Purchase App (Deadline: 17th February, 11:55 pm)

Web application based on MERN stack - MongoDB, Express.js, React.js, and Node.js.

Concerned TAs - Mayank, Prajwal, Puru, Sanjana, Eesha (Do not ask other TAs for any assignment specific queries/doubts). Also, **Moodle** is the recommended channel for communication.

Introduction

For this assignment, you are required to make a web app with:

- Frontend in **React.js**
- Backend using **Express.js** which implements a **REST** API
- Database in **MongoDB**

The app proposes to solve a major problem that the students of our college face. There are times when one wants to buy an item, only to find that buying them in bulk would make it much cheaper as opposed to buying a single unit. The app will have an option for sellers to host their products along with the minimum bulk dispatch quantity. Various customers can select from the listed products and order them with their own required quantity. When enough orders are placed for the product and bulk quantity requirements are met, the vendor can dispatch the order.

Requirements:

- There will be two types of users - Vendors and Customers.
- There must be a registration and login feature for both users. During registration, there would be the option to select between customer and vendor type. (10 marks)
- Use Cases of the Vendor:
    - Should be able to create a new product specifying the following: (5 marks)
        - Name of Product
        - Price of the Bundle
        - Quantity in the Bundle
    - Should be able to view all the current product listing done by him/her
        - There should be an option to take down a listing making sure that customers get their product status as canceled. (5 marks)

        - Once the product is ready to dispatch (i.e. when it has been ordered by sufficient people), it is removed from this view and becomes ready to dispatch. (5 marks)
    - Should be able to separately view all the orders that are ready to dispatch
        - Should have a button to dispatch the product which removes it from this view. (5 marks)
    - All dispatched orders should be displayed in another view with the reviews and ratings of each order. (5 marks)
- Use Cases of the Customer :
    - Should be able to search for the product he/she wants (Exact string matching would do)
        - All the vendors selling that product should be displayed along with their price and quantity remaining (5 marks)
        - Should be able to sort the search results either by price or quantity of items in bundle left or the rating of the seller (5 marks)
    - Should be able to select a product listed in the search results and place the order after specifying the quantity he/she desires (5 marks)
    - Should be able to separately view the status of all the products he/she has ordered and should contain:
        - Its dispatch status (10 marks)
            - Waiting (If not enough orders have been placed meeting the minimum bulk quantity requirement by the seller)
            - Placed (If the quantity requirements are met but is yet to get dispatched by the seller in his/her portal)
            - Dispatched (If the seller accepts the order in his/her portal)
            - Canceled (If the seller cancels the order in his/her portal)
        - In the case of Waiting State, the following also needs to be displayed/implemented (5 marks)
            - Quantity left for the order to get placed
            - Option to edit the order if not in the dispatched state
        - Should be able to rate the vendor once the order is placed. Average rating of the vendor must be displayed in the search results. (5 marks)
        - Should be able to give a product review along with a rating once the product has been dispatched. Clicking on a particular vendor in the search results should display their reviews. (10 marks)
- Basic Minimal UI for good user experience. (20 marks)

For example, consider a vendor who wants to sell 100 pens as a bulk product for Rs. 150. Different customers who want a pen can select this bundle and list the quantity that only he/she wants - one customer might want 3 pens, another wants 5 and so on. Once the requirement of 100 pens is done, the vendor is able to see this in another view and can choose whether or not to dispatch it. Once he dispatches it, this is removed from this view. Status on the customer dashboard changes accordingly.

Bonus:

- The seller having the option to upload the product images with the images getting displayed at the customer side while searching (10 marks)
- Better searching options like fuzzy matching (5 marks)
- Login through any social media (Facebook, Twitter, Google, etc) handles. (10 marks).

Additional Instructions

- Ensure that you have a proper README file that lists down how to run the code.
- You may use any npm package, as long as it's not off-the-shelf.
- In the final submission, ensure that you have **package.json** files. **Do not submit node_modules folder.**
- Cases of plagiarism will be given a straight zero. **DO NOT COPY ANY PART FROM ANY SOURCE** including your **friends**, **seniors** or the **internet**.
- Marks indicated for requirements assume proper working of frontend and handling of corresponding database operations.
- Please test your application thoroughly. Any errors occurring during the evaluation will be frowned upon.
- The use of Bootstrap/CSS templates are allowed. The same goes for Material UI.(Link)
- Follow good coding practices and good database designs.
- Exception Handling should be well implemented such that all the edge cases are taken care of.
- Please start ASAP. The deadline is a hard one, and won't be extended. Don't wait till the last minute to start working on it.

Deliverable

Submission must be in the following format:

```
<roll_no>
├─── backend/
├─── frontend/
└─── README.md
```

Where 'backend/' and 'frontend/' are directories with the Express.js and React app in it respectively. Zip this up and submit one **<roll_no>.zip** file.

Note:- There will be a 10% penalty for not following the submission format and straight zero for submitting corrupt zips.