NAME: Boobalaragavan P

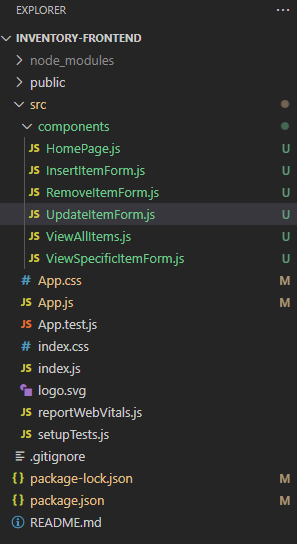
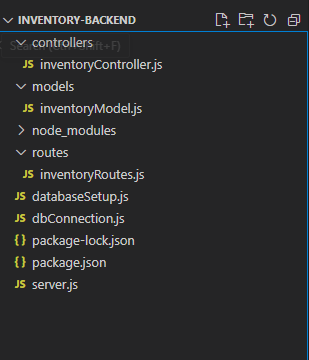
PHONE NUMBER: 7667992244

EMAIL ID: [boobalaragavan0520@gmail.com](mailto:boobalaragavan0520@gmail.com)

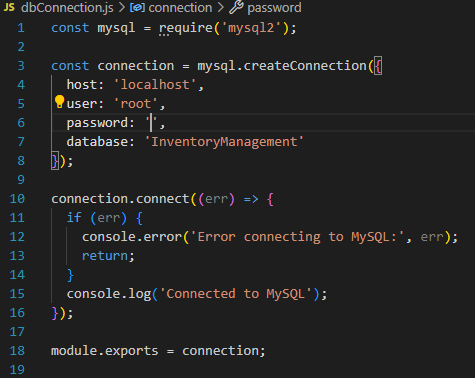
**INVENTORY MANAGEMENT SYSTEM**

1. **Project setup instructions :-**

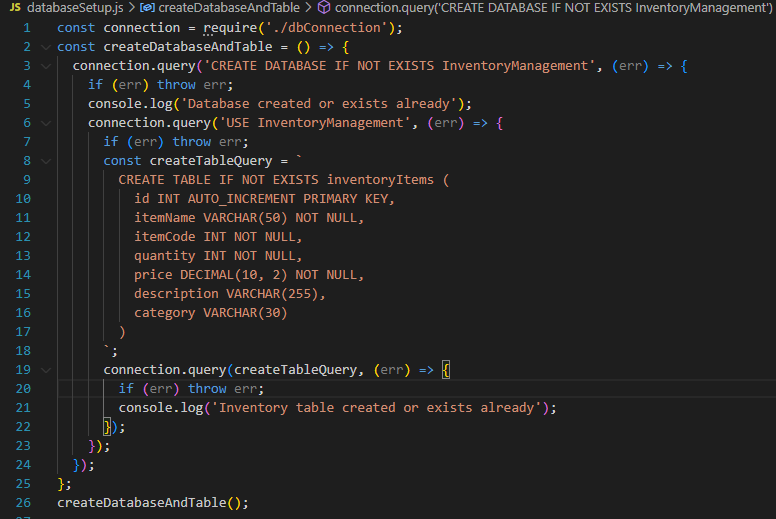
* Installation of Node
* Using the NPM packages installing express
* Using the NPM packages installing MySql2
* Installation of CORS – Cross Origin Resource Sharing
* Installation of React JS
* Installation of Axiom
* Installation of Material UI
* Installation of React-Router-Dom
* Folder Structure for frontend and backend

* Database Connection



* Database Setup

****

1. **System architecture :-**

BACK END APPLICATION

(Node Js)

DATABASE

(MySql)

FRONT END APPLICATION

(React Js)

1. **Interaction of components :-**
2. *Frontend (React JS with Material UI):*

* Developed the interface using react js and used material UI for styling purpose
* Interface allows users to interact with the system by performing CRUD operations on inventory items
* Communicates with the backend using Axios to make HTTP requests

1. *Backend (Node JS with Express):*

* Exposes Restful API Endpoints to handle requests from the frontend
* Uses MySql as the database to store the inventory data
* Business logics are developed for adding new inventory item, updating an existing inventory item by id, viewing all items available, viewing a specific inventory item by id, and removing an existing inventory item by id

1. *Database (MySql):*

* Stores inventory items in the database named “inventorymanagement” in the table named “inventoryitems”
* Each line item in the table contains fields such as auto incremented table Id, Item name, Item code, quantity, price, description and category.

1. **Workflow :-**
2. The user interact with the frontend to send the requests ( for example for adding, viewing, updating and removing items)
3. The frontend sends the request to the backend API
4. The backend process the request, interacts with the database, and returns a response

iv) The frontend displays the response to the user

1. **Api endpoint explanation :-**
2. *Retrieve all items:*

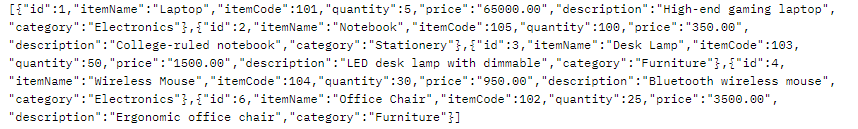
ENPOINT: GET/api/inventory

URL: <http://localhost:5000/api/inventory>

PURPOSE: Fetching all inventory items available in the database

REQUEST BODY: None

EXPECTED RESPONSE:



1. *Retrieve specific item:*

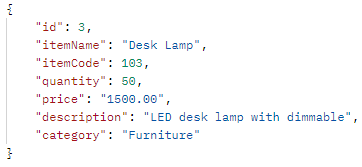
ENPOINT: GET /api/inventory/:id

URL: <http://localhost:5000/api/inventory/3>

PURPOSE: Fetching the specific inventory item based on the user input (id)

REQUEST BODY: None

EXPECTED RESPONSE:



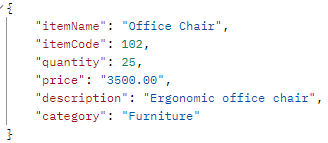
1. *Add a New Item:*

ENPOINT: POST /api/inventory/

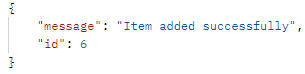
URL: <http://localhost:5000/api/inventory>

PURPOSE: Adding a new inventory item into the database

REQUEST BODY:



EXPECTED RESPONSE:



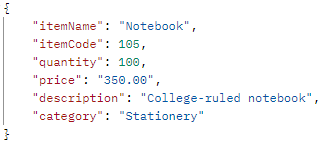
1. *Updating an existing item:*

ENPOINT: PUT /api/inventory/:id

URL: <http://localhost:5000/api/inventory/2>

PURPOSE: Updating the inventory item information based on the user input (id)

REQUEST BODY:



EXPECTED RESPONSE:



1. *Removing an item:*

ENPOINT: DELETE /api/inventory/:id

URL: <http://localhost:5000/api/inventory/5>

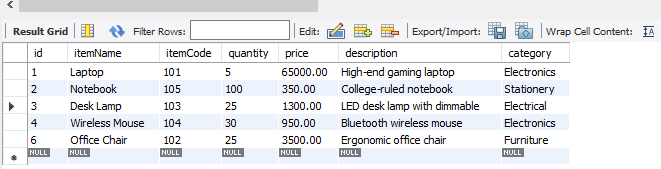
PURPOSE: Removing the specific inventory item

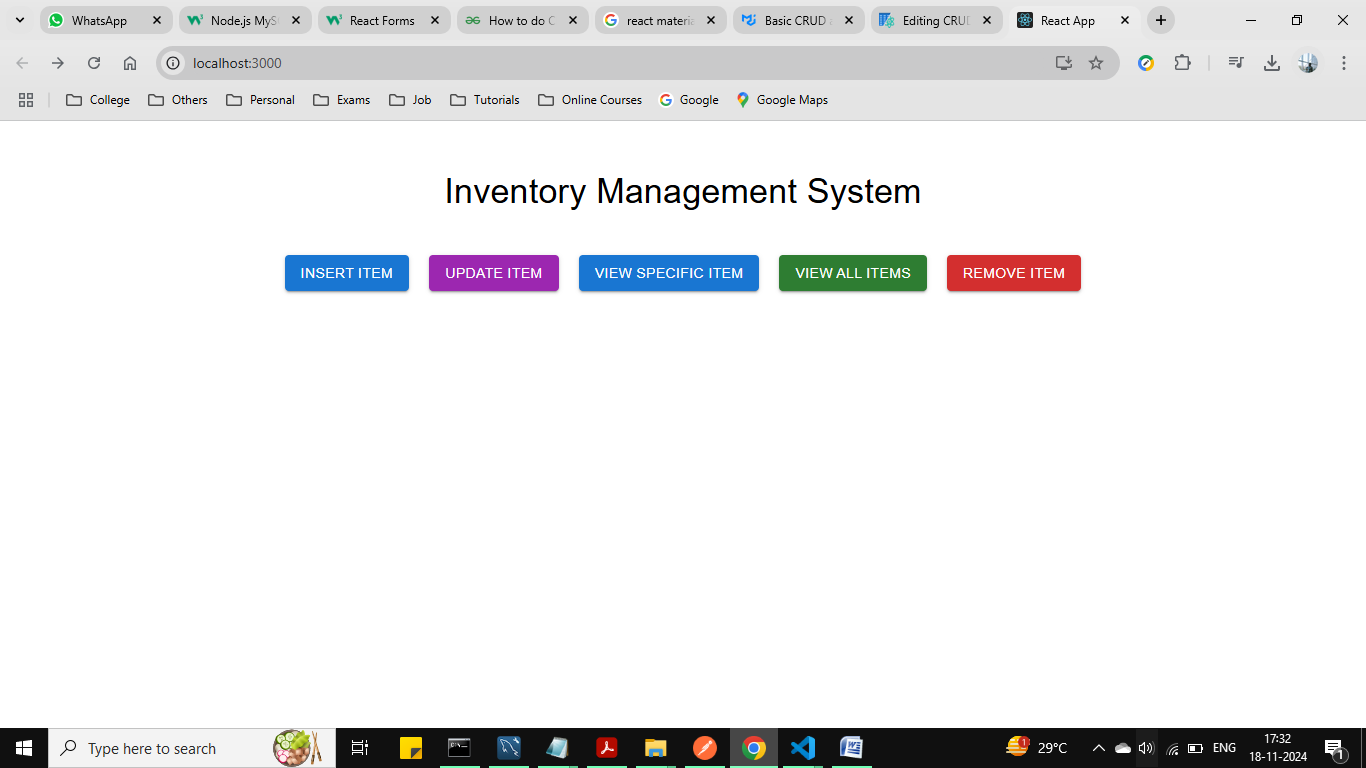
REQUEST BODY: None

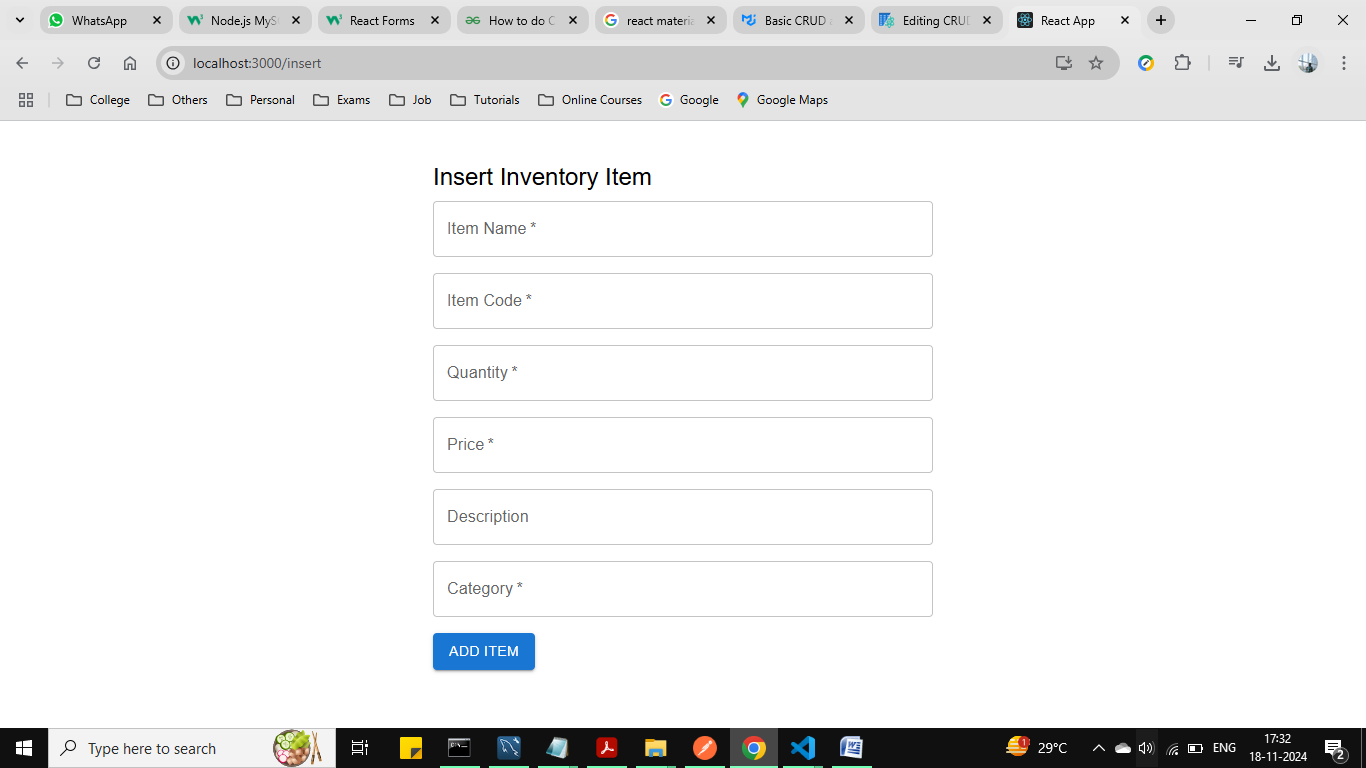
EXPECTED RESPONSE:

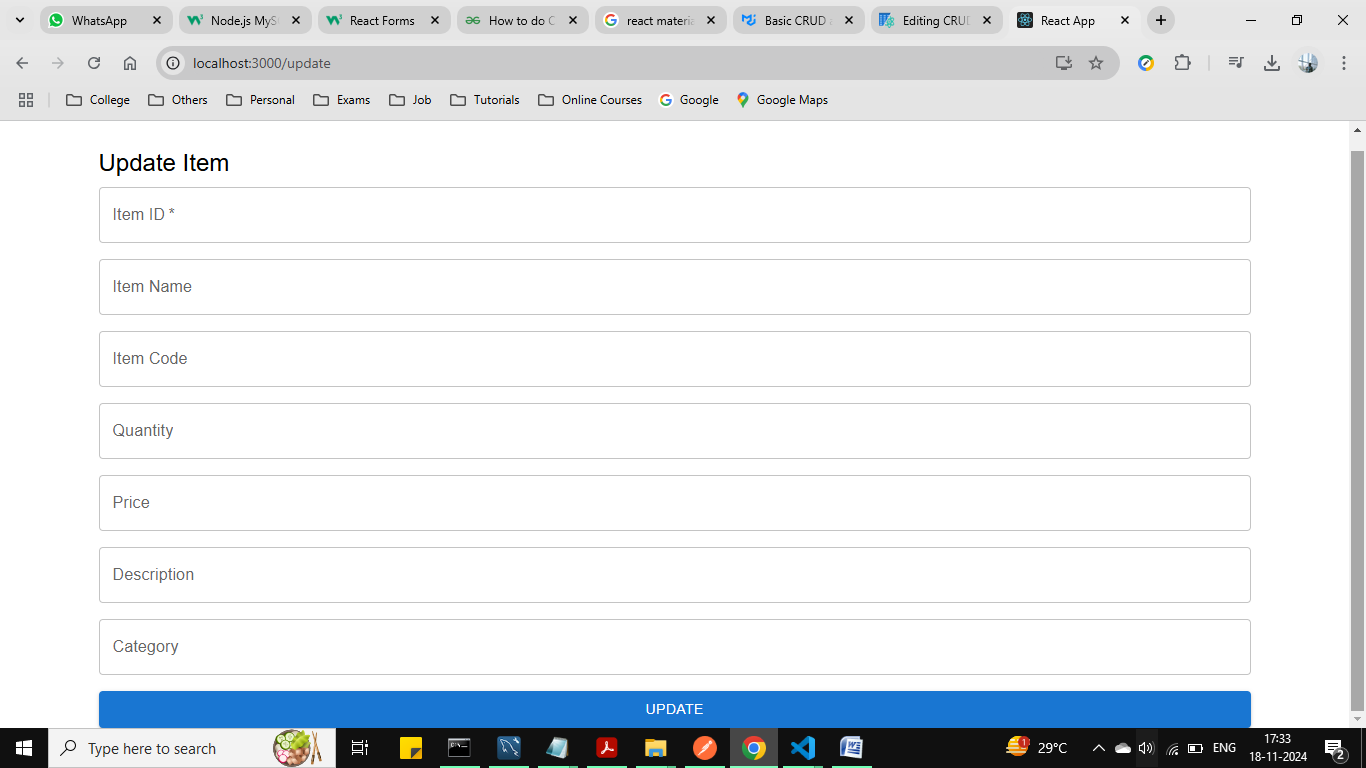


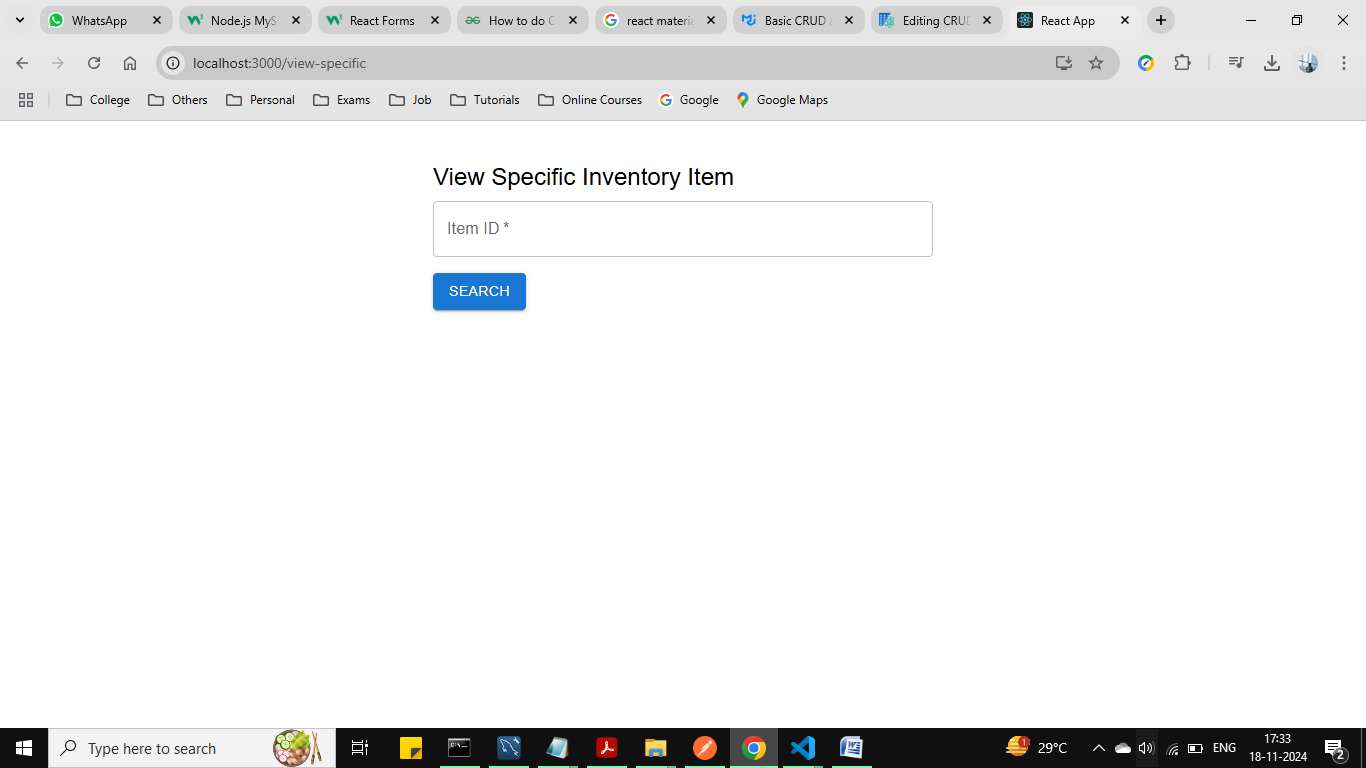
1. **Output Screenshots :-**

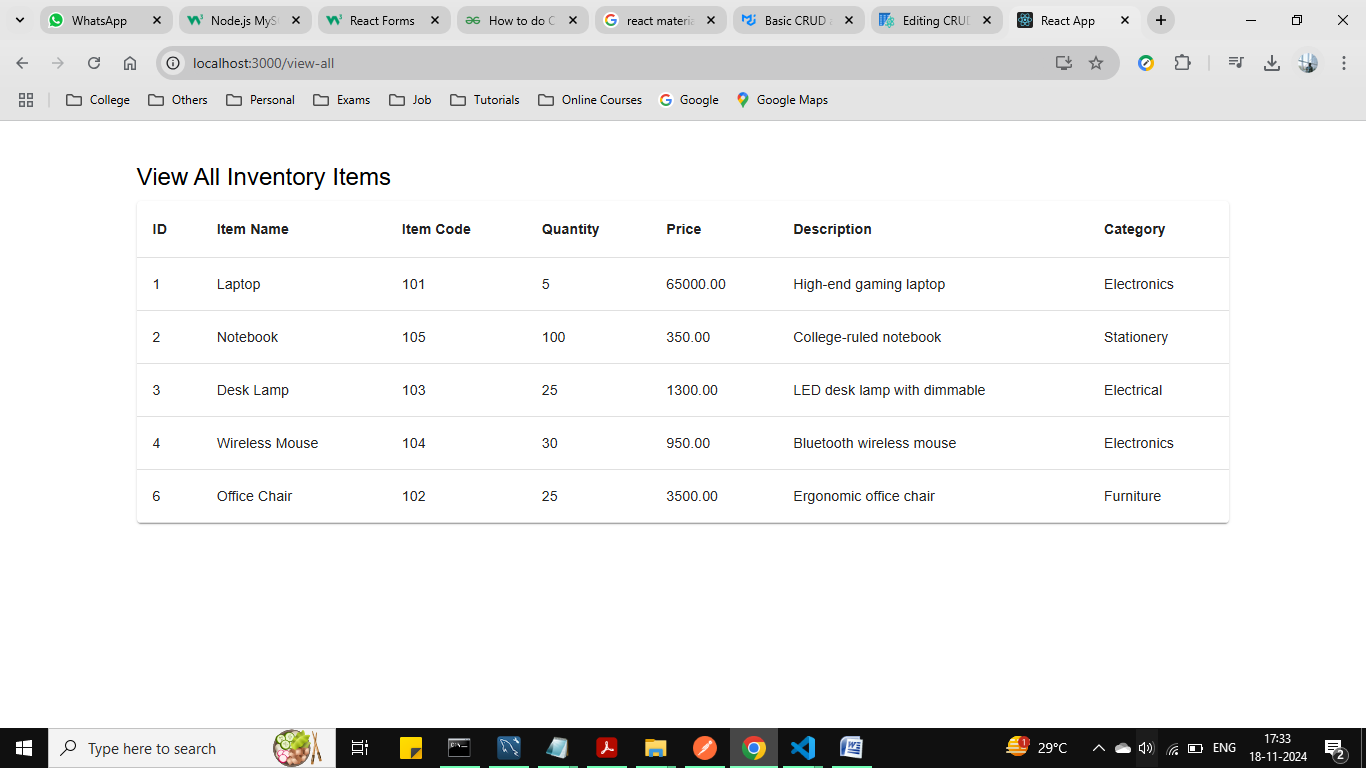


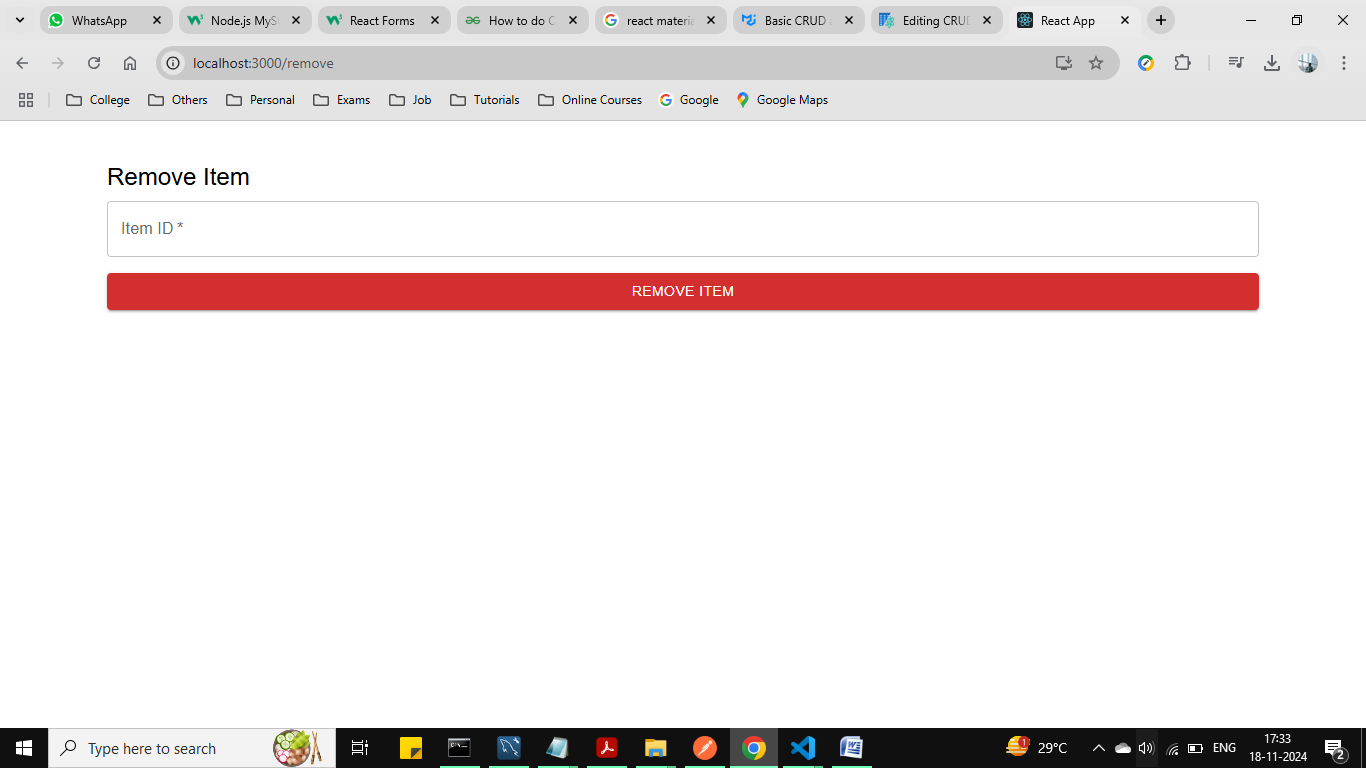
****

****

****

****

****

****