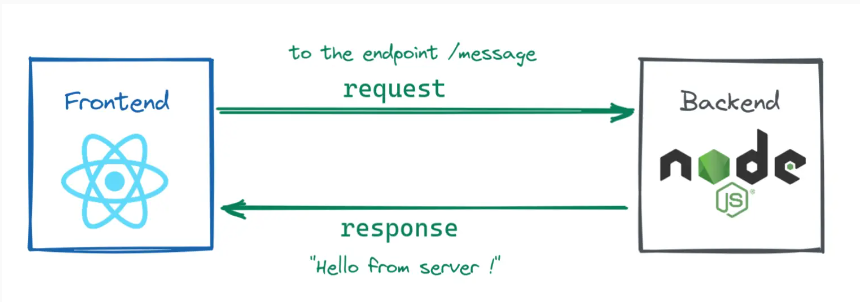
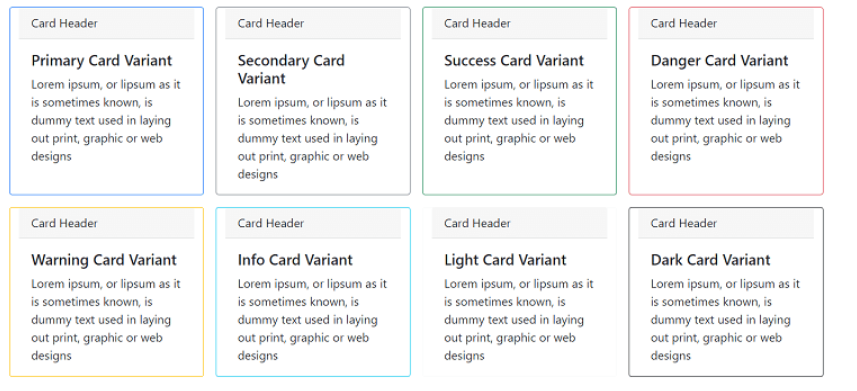
**UCS 2611 Internet Programming Lab**

**Exercise 9. Two Tier Architecture Application using React and NodeJs**

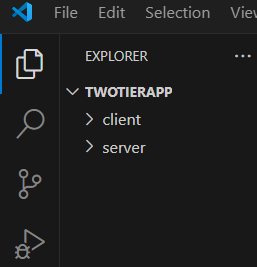
1. Develop two tier architecture application using React and NodeJs - Retrieve the information that store in a node and display it on the browser using two tier architectures.



Task: Store student details (such as name, rollno, contact-info) as array of objects in a node. Retrieve the details of each student from node and display on browser in the form of the cards as shown below image. In the place of Card Header and Card Content - display student roll and student name, contact-info.



Folder Structure:



1. **Create an empty folder and name as “twotierapp”**
2. **In this folder, create an empty folder and name as “server”**
   1. **Building the server using nodejs- backend:**
      1. Installing the dependencies
         1. Open a new terminal and move to server folder using –

***cd* server**

* + - 1. Initialize a Node.js project in it-

**npm init -y**

* + - 1. Let’s install the required dependencies for our application.

**npm install express cors**

* + - 1. For development purposes, we will install nodemon as a development dependency.

**npm install nodemon --save-dev**

* + - 1. create a new file and name it as “Server.js”- write backend logic

**Sample Code: (Server.js)**

const express = require("express");

const cors = require("cors");

const app = express();

app.use(cors());

app.use(express.json());

app.get("/message", (req, res) => {

  res.json({ message: "Hello from server!" });

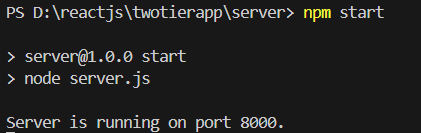
});

app.listen(8000, () => {

  console.log(`Server is running on port 8000.`);

});

* + - 1. **Run the code in terminal**

****

1. **Building the client-side application using React (front-end)**

* First, we will create a React application. Open your terminal and run the following command to create a React application.
* **npx create-react-app client**
* Go to App.Js and write client-side logic for fetching the data from node and to render on the browser.

**Sample code: (App.js)**

import React, { useState, useEffect } from "react";

import "./App.css";

function App() {

  const [message, setMessage] = useState("");

  useEffect(() => {

    fetch("http://localhost:8000/message")

      .then((res) => res.json())

      .then((data) => setMessage(data.message));

  }, []);

  return (

    <div className="App">

      <h1>{message}</h1>

    </div>

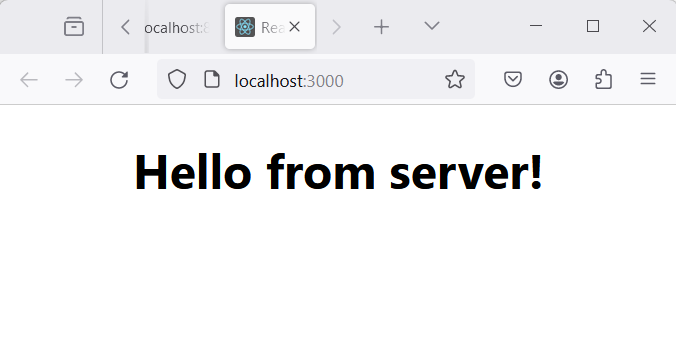
  );

}

export default App

**Run the App.js in terminal: npm start**

**Output For Above Sample Code:**

****

**Best Practices to be followed:**

1. Design before coding

2. Incremental coding

3. Usage of proper naming convention

4. Usage of Comments to the code

5. Indentation of code