Assignment - 7.3

Frontend & Backend Integration

→ In web development, the frontend and backend are distinct layers of an application that communicate to provide a seamless user experience. The frontend is responsible for the user interface and user interface, while the backend handles business logic, database interactions and server side processing. To integrate these two layers, developers often use API's, specially RESTful APIs, to enable communication between the client (frontend) & the server (backend).

→ **Restful API** -

A restful API is an API that adheres to the principles of REST (Representational State Transfer). REST is an architectural style for designed networked applications, relying on a stateless, client-server communication protocol, typically HTTP.

→ **Principles of REST**-

1. **Client-Server Architecture** - Separates the user interface concerns from data storage concerns, improving the scalability & probability of an application.

2. **Statelessness** - Each request from a client to a server must contain all the information needed to understand & process the request.

3. **Cachebility** - Responses must defines themselves as cachebility or non-cachable to improve client-side performance.

4) Uniform Interface - Simplifies & decouples the architecture, allowing each part to evolve independently.

5) Layered System - The client cannot tell whether it is connected directly to the end server or an intermediary, improving scalability & security.

6) Code on Demand - Servers can extend client functionality by transferring executable code.

HTTP Methods -

1) GET: Retrieve data from server.
2) POST: Send data to server to create a new resource.
3) PUT: Update an existing resources on server.
4) DELETE: Remove a resource from server.
5) PATCH: Apply partial modifications to a resources.

⇒ Making API calls from Frontend -

① "Fetch" -
```
fetch('http://api.example.com/data')
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.log.error('error', error));
```

② "Axios" -
```
axios.get('http://api.example.com/data')
  .then(response => {
      console.log(response.data);
  })
  .catch(error => {
      console.error('Error', error);
  });
```

→ Create the Server -

① app.get('/api/data', (req, res) => {
    res.json(data); });

② app.listen(port, () => {
    console.log('Server running at ${port}');
});