FSD ASSIGNMENT - 7

By-Yashvardhan Tekavade
PB-15 Batch-1
Panel-1 TY-CSF

Aim: Develop a full-stack web application using MERN stack to perform CRUD operations.

Objectives:

1. To develop full-stack web projects using the MERN stack.
2. To learn database connectivity using fetch api.
3. To perform insert, update, delete and search operations on d

Theory

1. What is the MERN stack?

The MERN stack is a popular software development stack that is used to build full-stack web applications. MERN stands for MongoDB, Express.js, React, and Node.js—each representing a different component of the stack.

MongoDB: A NoSQL database that stores data in a flexible, JSON-like format called BSON (Binary JSON). MongoDB is known for its scalability and flexibility, making it suitable for handling large volumes of data and diverse data types.

Express.js: A web application framework for Node.js that simplifies the process of building robust and scalable web applications. Express.js provides a set of features for building web and mobile applications, including routing, middleware support, and an easy-to-use API.

React: A JavaScript library for building user interfaces. Developed by Facebook, React allows developers to build reusable UI components that update efficiently and in a predictable way. It is commonly used for creating interactive and dynamic user interfaces.

Node.js: A JavaScript runtime that allows developers to run server-side JavaScript. Node.js is known for its non-blocking, event-driven architecture, making it efficient for building scalable network applications. It is commonly used to build the server side of web applications.

2. Use of Fetch API.

The Fetch API is a modern web standard for making HTTP requests in web browsers and Node.js environments. It provides a more powerful and flexible alternative to the older XMLHttpRequest.

Here are some common use cases and features of the Fetch API:

Making HTTP Requests:

- The primary purpose of the Fetch API is to make HTTP requests to servers. It supports various HTTP methods such as GET, POST, PUT, DELETE, etc.
- It returns a Promise that resolves to the Response to that request, allowing for easy handling of asynchronous operations.

Handling Responses:

- The Response object returned by Fetch provides methods to access and handle the response data, including text, JSON, or Blob.
- It also allows checking the response status, headers, and other metadata.

Sending Data:

- Fetch enables sending data in the request body, which is useful for methods like POST and PUT. This can include JSON, FormData, or other types of data.
- It provides options to customize the request headers, content type, and other parameters.

Working with JSON:

- Fetch makes it easy to work with JSON data. You can use the json() method on the Response object to extract JSON content from the response.

Example of a simple Fetch API request:

```
fetch('https://api.example.com/data')
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

FAQ:

1. What makes the MERN stack the fastest-growing tech stack?

The MERN stack's rapid growth can be attributed to several factors:

JavaScript Unity: Using JavaScript across the entire stack (MongoDB, Express.js, React, Node.js) promotes code reusability and reduces context switching.

Full-Stack Capabilities: MERN offers a comprehensive solution for both front-end and back-end development, enabling developers to build end-to-end applications seamlessly.

React's Component-Based Approach: React's component-based architecture simplifies UI development by breaking it into modular and reusable components.

Rich Ecosystem: The MERN stack benefits from a robust ecosystem of libraries and tools, facilitated by the Node Package Manager (NPM).

Community Support: Strong communities around each component foster collaboration, knowledge sharing, and resource availability.

Sample Problem Statements:

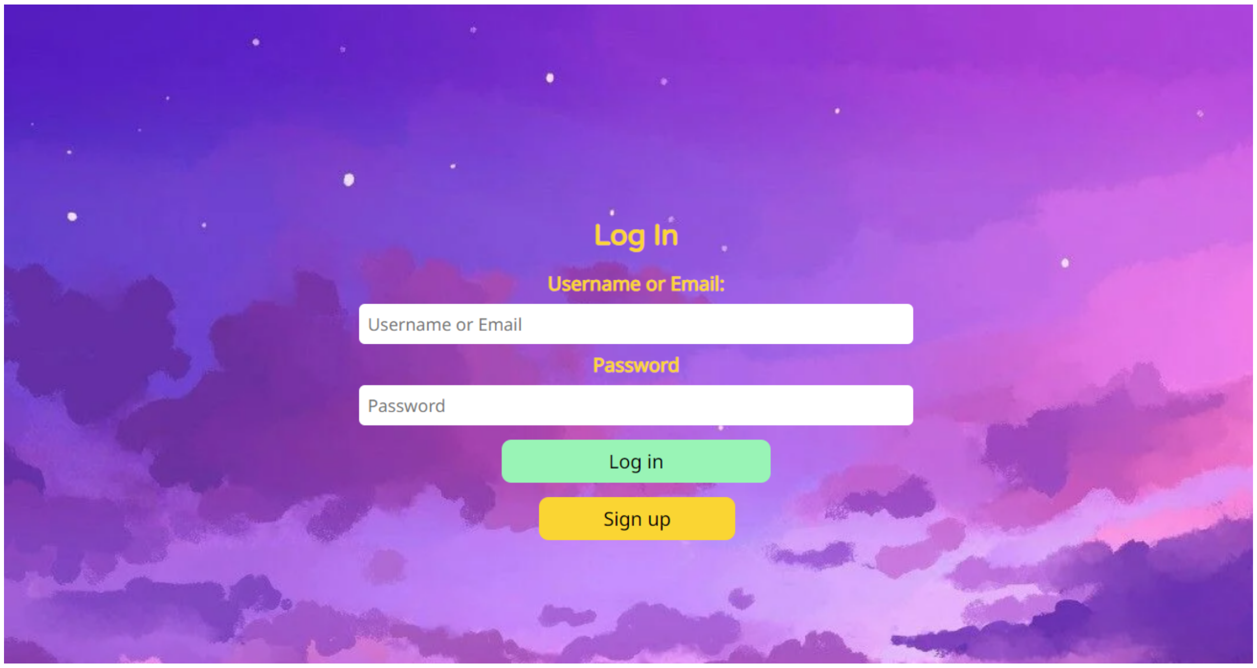
CRUD Operations using MERN stack.

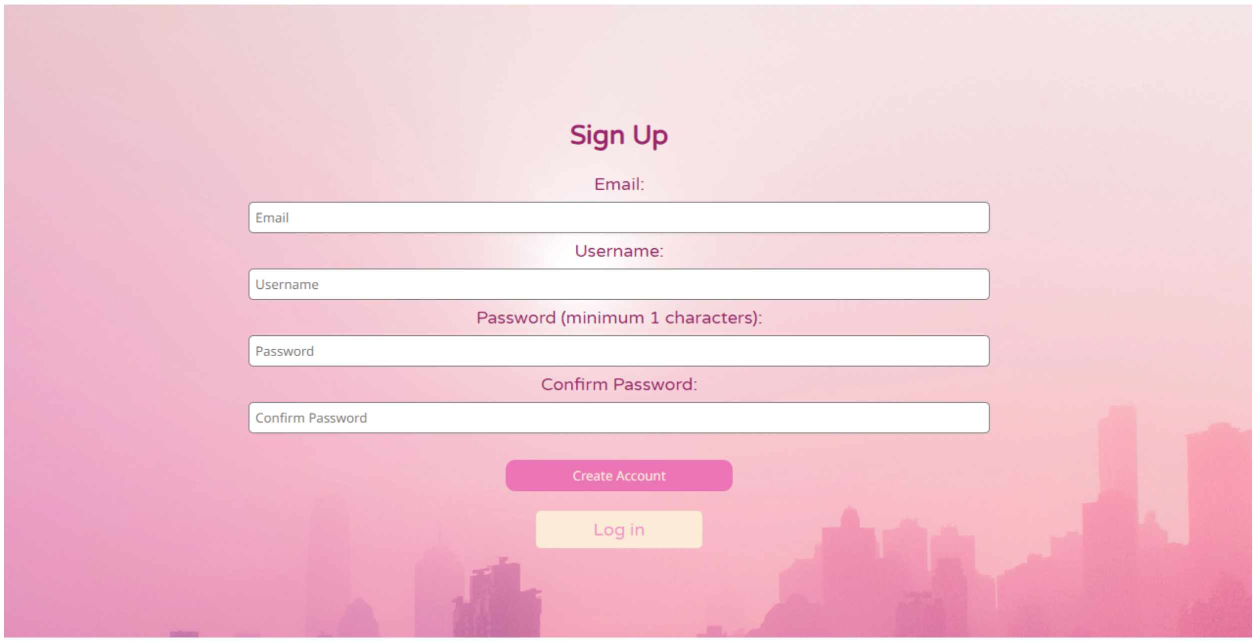
3. Student can create a React form or use existing/ implemented HTML form for Employee Management System with the fields mentioned: Employee name, Employee ID, Department_name, Phone number, Joining Date and perform the following operations

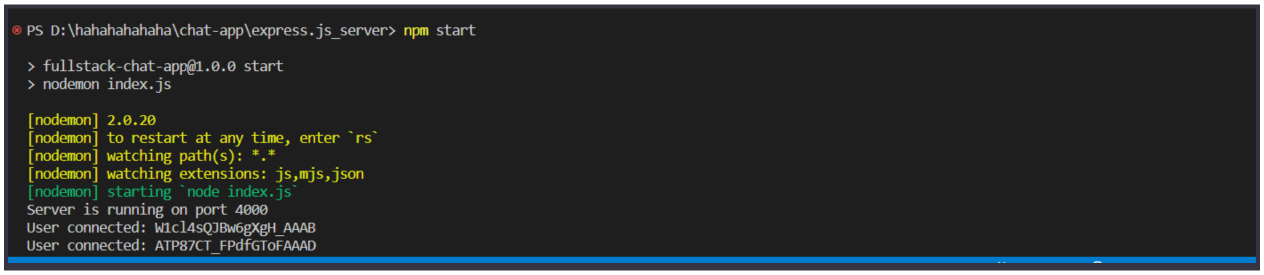
1. Insert Employee details -Employee name, Employee ID, Department_name, Phone number, Joining Date
2. Delete the Employee records based on Employee ID
3. Update the Employee details based on Employee ID- For example students can update Employee details based on searching the record with Employee ID.
4. Display the Updated Employee details or View the Employee Details records in tabular format.

Help Link:

https://www.mongodb.com/languages/mern-stack-tutorial
Output: Screenshots of the output to be attached.







```
PS D:\hahahahahaha\chat-app\express.js_server> npm start

 > fullstack-chat-app@1.0.0 start
 > nodemon index.js

 [nodemon] 2.0.20
 [nodemon] to restart at any time, enter `rs`
 [nodemon] watching path(s): *.*
 [nodemon] watching extensions: js,mjs,json
 [nodemon] starting `node index.js`
 Server is running on port 4000
 User connected: W1cl4sQJBw6gXgH_AAAB
 User connected: ATP87CT_FPdfGToFAAAD
```