Thought Note: To-Do List Application

- This project involves creating a To-Do list application using a combination of backend and frontend technologies. The backend is implemented using Node.js and Express.js, while the frontend is built with React.

- The application allows users to enter tasks through an input box, which are then stored in the node-persist storage. The entered tasks are displayed below in the tasks list section. The storage is initialized with a specified directory for data persistence.

- One interesting aspect of this project is the use of node-persist for storing data. It provides a simple and efficient solution for persisting data on the server-side. The storage.clear() function is utilized to delete old data when the application is restarted.

- The frontend component ToDoForm handles the input and display of tasks. It consists of an input box where users can enter tasks and a submit button to add them to the list. The todo state variable is used to store the value of the input field, and the setTodo function is used to update its value.

- On form submission, the onsubmitHandler function is triggered. It prevents the default form submission behavior, sends a POST request to the backend API endpoint /todo, and clears the input field. Upon receiving a response, it calls the getTodos function to fetch and update the list of tasks.

- The getTodos function sends a GET request to the backend /todo endpoint to retrieve the list of tasks. It updates the itemList state variable with the received data. The useEffect hook is used to call the getTodos function when the component mounts.

- The useEffect hook is also used to handle the cleanup process when the component unmounts. It sends a DELETE request to the /todo endpoint to clear the storage and delete all existing tasks.

- The rendered JSX elements include a heading for the application, the input box, and an ordered list (<ol>) to display the tasks. The itemList state variable is mapped over to render individual ToDoList components, passing each task as a prop. The ToDoList component is responsible for rendering a single task as an <li> element.

- Overall, this project combines backend and frontend technologies to create a complete To-Do list application. It demonstrates the use of storage for data persistence and showcases the seamless integration between the frontend and backend components. By completing this project, you'll gain hands-on experience in building a practical application with essential features.