What Is Express JS ?

ANS ➔ Express is a fast, assertive, essential and moderate web framework of Node.js. You can assume express as a layer built on the top of the Node.js that helps manage a server and routes. It provides a robust set of features to develop web and mobile applications.

features of Express :

- *It can be used to design single-page, multi-page and hybrid web applications.*
- *It allows to setup middlewares to respond to HTTP Requests.*
- *It defines a routing table which is used to perform different actions based on HTTP method and URL.*
- *It allows to dynamically render HTML Pages based on passing arguments to templates.*
- *Ultra fast I/O*
- *Asynchronous and single threaded*
- *MVC like structure*
- *Robust API makes routing easy*

Create localhost server using express ?

ANS ➔ *Install Express with npm. Define routes and functionality in a JavaScript file. Start the server by listening on a specified port. Test functionality by navigating to localhost in a web browser*.

Create basic "Hello world" app using node and express ?

And ➔ *To create a basic "Hello world" app using Node.js and Express, first, initialize a new Node.js project. Then, install Express.js using npm. Next, create a JavaScript* ***file (example : ➔ app.js)*** *and define an Express server on a specified port. Set up a route that responds with "Hello world" when accessed. Finally, start the server using* **app.listen()** *method. Accessing* **http://localhost:3000** *in a web browser will display the message "Hello world".*

Create weather app. ? (Use Hbs) (Get country, state, city using api) (Get weather data by city)

Ans ➔ *To create a weather app using HBS, Node.js, and Express.js, integrate APIs to fetch country, state, and city data. Use APIs like GeoNames or OpenCage to obtain location information based on user input. Then, utilize a weather API like OpenWeatherMap to retrieve weather data for the specified city. Implement routes to handle these API requests, fetching location and weather data asynchronously. Render the fetched data using HBS templates to display detailed weather information to the user. Ensure error handling for API requests and provide a user-friendly interface for inputting city names. Start the Express server to enable users to access the weather app, input a city, and retrieve weather details effortlessly.*

Create world clock app.

Ans ➔ *Create a world clock app with Node.js and Express, integrating HBS for templating. Fetch time data from an external API for different cities worldwide. Use HBS templates to dynamically render the data. Implement user-friendly features such as sorting and adding/removing clocks. Ensure proper error handling and responsiveness. Test thoroughly for functionality. Users access the app via a web browser to view current times and customize clock display. With Node.js, Express, and HBS, build a versatile world clock app tailored to user preferences and needs.*

What is MongoDB.

Ans ➔ *MongoDB is a NoSQL database that stores data in flexible JSON-like documents, providing scalability and ease of use for diverse applications. Its schema-less architecture allows for easy adaptation to changing data requirements, making it popular for modern web and mobile applications.*

What is difference between mongo DB and SQL

Ans ➔ *MongoDB stores data as JSON-like documents, offering flexibility. SQL databases use structured tables with predefined schemas. MongoDB suits unstructured data, while SQL is preferable for structured data and complex queries*.

Create database for online shopping app.

Ans ➔ For an online shopping app, create a relational database with tables for products, customers, orders, order items, and transactions. The products table stores details like ID, name, price, and quantity. Customers table holds customer information such as name, email, and address. Orders table records each order with order ID, customer ID, and status. Order items table links products to orders, containing details like quantity and price. Lastly, transactions table logs payment transactions associated with orders. This database structure allows for efficient management of products, customer orders, and payment transactions, facilitating smooth operation of the online shopping platform.

Create Require collections for online shopping app and documents?
(User) (Product category) (Product) (Order) (Review)

Ans ➔

User : *The User collection stores user information such as username, email, password, address, and payment details, facilitating user registration and authentication within the app.*

Product Category : *The Product Category collection categorizes products, storing category names and descriptions, enabling efficient organization and navigation within the online shopping app's inventory for a seamless user experience.*

Product : *The Product collection stores details of products available for purchase in the online shopping app, such as name, description, price, and quantity available. Each product document may reference a Product Category for organizational purposes, facilitating efficient browsing and management of the app's inventory.*

Order : *The Order collection tracks orders placed by users in the online shopping app, recording details such as user ID, products ordered, total price, order date, and status. It facilitates order processing and management within the app.*

Review : The Review collection stores user feedback for products in the online shopping app, including ratings, comments, and submission dates. It enhances product evaluation and assists users in making informed purchasing decisions.

Write command to show all data from product collections and sort in ascending order.

Ans :→ *To display all data from the Product collection and sort it in ascending order, execute the MongoDB command: db.Product.find().sort({ fieldName: 1 }). Replace fieldName with the desired sorting parameter, such as name or price. This command retrieves and arranges documents from the Product collection based on the specified field, ensuring organized data presentation for effective navigation and user interaction within the online shopping app.*

Update product price for particular product.

Ans : →

To update the price of a specific product in the Product collection, use MongoDB's updateOne() method. Provide the product's ID in the filter parameter and set the new price using the $set operator.

db.Product.updateOne ({ _id: ObjectId("product_id") }, { $set: { price: new_price } })

Replace "product_id" with the ID of the product to update and new_price with the desired price value. This command will modify the price of the specified product in the Product collection, ensuring accurate pricing information for the online shopping app.

Write command to delete particular document and collection.

Ans : → *To delete a specific document from a MongoDB collection, use the deleteOne() method with a filter specifying the document to remove.*

```
db.collection.deleteOne ( { _id: ObjectId("document_id") } )
```

Replace "document_id" with the ID of the document to delete. This command ensures precise removal of the specified document.

To delete an entire collection, employ the drop() method.

```
db.collection.drop()
```

Executing this command eliminates the entire collection, including all its documents. Exercise caution when dropping a collection, as it results in permanent data loss. Always confirm the necessity of deletion operations to avoid accidental data loss.

Create Rest API (get, post, delete, put, patch) for Task management ?

Ans : → *To create a REST API for Task management, use Express.js with Node.js. Define routes for each HTTP method:*

GET: Retrieve tasks

POST: Create a new task

DELETE: DELETE A TASK

PUT: Update a task

PATCH: Partially update a task

For example, a GET request to /tasks retrieves all tasks, while a POST request to /tasks creates a new task. Similarly, DELETE, PUT, and PATCH requests modify tasks based on their IDs. Implement corresponding controller functions to handle these requests and interact with a database or data source to manage tasks efficiently.

Create Rest API (get, post, delete, put, patch) for online shopping application ?

Ans : → *To create a REST API for an online shopping application, we'll use Express.js with Node.js. The API supports various HTTP methods:*

GET: Retrieve products, orders, and user information.

POST: Add new products, create orders, and register users.

DELETE: REMOVE PRODUCTS, CANCEL ORDERS, AND DELETE USER ACCOUNTS.

PUT: Update product details, modify order status, and edit user information.

PATCH: Partially update product attributes, change order details, and adjust user data.

Each method corresponds to specific endpoints, such as /products, /orders, and /users, with corresponding controller functions handling requests to interact with the database and manage application data effectively.