Q1. Create Rest API (get, post, delete, put, patch) for Task management.

🡺const express = require('express');

const app = express();

app.use(express.json());

let tasks = [

{ id: 1, title: 'Task 1', description: 'Description 1', completed: false },

{ id: 2, title: 'Task 2', description: 'Description 2', completed: true }

];

// GET all tasks

app.get('/tasks', (req, res) => {

res.json(tasks);

});

// GET a task by ID

app.get('/tasks/:id', (req, res) => {

const task = tasks.find(t => t.id === parseInt(req.params.id));

if (!task) return res.status(404).send('Task not found');

res.json(task);

});

// POST a new task

app.post('/tasks', (req, res) => {

const newTask = {

id: tasks.length + 1,

title: req.body.title,

description: req.body.description,

completed: req.body.completed || false

};

tasks.push(newTask);

res.status(201).json(newTask);

});

// PUT (update) a task by ID

app.put('/tasks/:id', (req, res) => {

const task = tasks.find(t => t.id === parseInt(req.params.id));

if (!task) return res.status(404).send('Task not found');

task.title = req.body.title;

task.description = req.body.description;

task.completed = req.body.completed;

res.json(task);

});

// PATCH (partial update) a task by ID

app.patch('/tasks/:id', (req, res) => {

const task = tasks.find(t => t.id === parseInt(req.params.id));

if (!task) return res.status(404).send('Task not found');

if (req.body.title) task.title = req.body.title;

if (req.body.description) task.description = req.body.description;

if (req.body.completed !== undefined) task.completed = req.body.completed;

res.json(task);

});

// DELETE a task by ID

app.delete('/tasks/:id', (req, res) => {

const taskIndex = tasks.findIndex(t => t.id === parseInt(req.params.id));

if (taskIndex === -1) return res.status(404).send('Task not found');

const deletedTask = tasks.splice(taskIndex, 1);

res.json(deletedTask);

});

const port = process.env.PORT || 3000;

app.listen(port, () => console.log(`Server running on port ${port}`));

Q2. What is Middleware in Express.js?

🡺Middleware in Express.js refers to functions that have access to the request (req), response (res), and the next() middleware function in the application's request-response cycle. Middleware functions can perform tasks like logging, authentication, handling errors, parsing request bodies, and more.

app.use((req, res, next) => {

console.log('Request URL:', req.originalUrl);

next();

});

Q3 . What is json web tokens?

🡺JSON Web Tokens (JWT) are a compact, URL-safe means of representing claims to be transferred between two parties. JWTs are commonly used for authentication and authorization. The token consists of three parts:

1. **Header**: Contains metadata such as the signing algorithm and token type.
2. **Payload**: Contains claims, which are statements about an entity (typically, the user) and additional metadata.
3. **Signature**: The signature is created by signing the header and payload with a secret key or a public/private key pair

Q4. What is different between encryption and hashing.

* 🡺**Encryption**: The process of converting data into a cipher (unreadable format) so that only authorized parties can decrypt it back to its original form. Encryption is reversible if you have the right key. It is typically used for data security and confidentiality. Examples: AES, RSA.
* **Hashing**: A one-way process that converts data into a fixed-length hash (unique string) and is irreversible. Hashing is primarily used for data integrity and verifying information. Examples: MD5, SHA-256. Hashing is used in scenarios like password storage (hashing passwords) and verifying file integrity (hashing files to compare).
* **Encryption**: Reversible (with the key).
* **Hashing**: Irreversible (one-way).