Theory Assignment -1

Name:- Aryya Rana

Roll No:- 66

Sem:- 7

Subject:- Application Developement using Full Stack (701)

Email:- aryarana.msc it20@vnsgu.ac.in

Q1) NodeJS :- Introduction, features, execution archit

A)

→ Introduction:-

- Nodejs is an open source, cross-platform runtim environment built on chrome's v8 Javascript engine. It allows developers to execute java code outside the browser, making it poss to build server-side applications.

- Node.js enables developers to create scalable, efficient, and real-time application, and it has gained immense popularity in the web developme community due to its non-blocking, event-driven achitecture.

⇒ Features of Nodejs :-

1) Asynchronous and Non-blocking :-

- Nodejs uses an event-driven, non-blocking I/O model, allowing it to handle multiple concurrent connections efficiently without getting blocked by time-consuming operations.

2) Single - Threaded, Event-loop Architecture:-

- Node.js operates on a single-threaded event loop, which enables it to handle asynchronous operation efficiently.

3) Fast Execution:-

- Node.js is built on the V8 engine, which compiles Javascript code into highly efficient machine code, leading to faster execution compared to traditional interpreted languages.

4) NPM (Node Package Manager):-

- NPM is a powerful package manager that comes bundled with node.js.

5) Cross-platform:-

- Node.js is compatible with various operating systems, including: Windows, macOS and Linux making it a versatile choice for building applications that can run on different platform.

6.) Scalability:-

- Due to its non-blocking nature, Node.js is highly scalable and can handle a large number of concurrent connections with relativity low resource consumption.

=> Execution Architecture:-

1) Event loop:-

The event loop is the heart of Node.j responsible for handling all asynchronous ope

2) Event Queue:-

- The event queue holds all the callbacks waiting to be executed. When an asynchrono operation completes, its callback is placed in th event queue.

3) Callback Functions:-

- Callback funcn are used to handle the results of async. opern.

4) V8 engine :-

- Node.js uses the V8 JS engine to execute JS code.

5) Libuv :-

- Libuv is a library that provides the event loop and handles I/O operations, allowing Node.js to be cross-platform.

Q2) Note on modules with eg.

A:)

→ In Node.JS, modules are used to encapsulate reusable pieces of code, making it easier to organize and maintain large applications.

→ A module in Node.js is a self-contained piece of code that is organized, promoting code reusability and reducing namespace collisions.

⇒ Creating a module:-

```
const calculate = (w,h) => {
    return w*h;
};
module.exports = calculate;
```

⇒ Using a Module:-

```
const cal = require('./rectangle');
const width = 5;
const height = 10;
const area = calculate(width, height);
console.log(area);
```

Q3) Note on package with example;

A)

- In Node.js a package refers to a collection of modules, libraries and other resources bundled together and published on the npm registry.
- Downloading a package is very easy.
- Here, I want to download a package called "upper-case". So in Command Line interface and download NPM.

C:\Users\MyPc> npm install upper-case.

- NPM creates a folder named "node-modules" where the package will be placed.
- Now folder structure will be like this.

C:\users\MyPc\node-modules\upper-case

⇒ Using a Package:-

- Once the package is installed, it is ready to use.
- Include the 'upper-case' package the same way you include any other mod

```
var uc = require('upper-case');
```

- Create a Node.js file that will conver "Hello world!" into upper case letters:-

```
var http = require('http');
var uc = require('upper-case');

http.createServer(function (req, res) {
    res.writeHead(200, {'Content-Type': 'text-
    res.write(uc.upperCase("Hello world!"));
    res.end();
}).listen(8080);
```

Q.h) - Use of package.json and package-lock.json.

A)

→ Both 'package.json' and 'package-lock.json' are essential files in Node.js project. They serve different purposes but work together to manage project dependencies and ensure consistent package installations across different environments.

→ 'package.json':—

- The 'package.json' file is the manifest file for a node.js project. It contains introduction metadata about the project, including its name, version, description, entry point, scripts, etc.

1) ① Dependency Management:—

- The dependencies and devDependencies sections in 'package.json' list the packages required for the project to run and for development purposes, resp.

2) Scripts :-

- You can define custom scripts in the scripts section of package.json. These scripts can be executed using npm run <script-name>. Common scripts include 'start' to run the app, and 'test' to execute unit tests.

3) Meta data :-

- The 'package.json' file contains essential information about the project, such as author's name, project description, license, repository, etc.

- package-lock.json :-

- The 'package-lock.json' file is automatically generated by npm when dependencies are installed or updated.

Q5) Nodejs Packages:-

A)

→ In the Node.js ecosystem, packages are coll. of modules, libraries, and resources that developers can use to enhance their application

→ Node.js packages provide func. for various purposes, ranging from web dev. and server-side tasks to command-line utilities and more.

→ Web Frameworks:-

- Packages - like express.js, koa and Hapi are popular web frameworks, that simplify the process of building web applications and apis by providing routing, etc.

→ Utility Libraries:-

- Packages like Lodash, Ramda, and Underscore.js provide utility func. that assist with tasks like data manipulation, validation and functional prog.

→ Database Libraries:-

- Packages like Mongoose and Sequelize provide easy-to-use abstractions for working with databases and ORM.

→ Authentication and Security:-

Packages like Passport.js and bcrypt offer solutions for authentication and password hashing to enhance app security.

→ Template engines:-

Packages like EJS, handlebars, and pug enable developers to generate dynamic HTML content easily.

→ HTTP clients:-

Packages like Axios provides tools for making http requests allowing Node.js applications to interact with APIs.

→ Real-time communications:-

- Packages like socket.IO facilitate real-time communication between clients and servers using WebSockets.

→ File System Utilities:-

- Packages like fs-extra and glob provide additional functionalities and ease of use for working with the file system.

→ Logging:-

- Packages like Winston and Bunyan provide flexible logging soln for Node.js appn

Q5) NPM introduction and commands with its use:-

A)

→ npm is the default package manager for Node.js and it is one of the largest software registeries in the world.

- It allows developers to easily install, manage and distribute Node.js packages to be used in their projects.

=> npm init:-

- This command initializes a new Node.js pro and creates a 'package.json' file. It prom you to enter details about the proje such as: name, version, etc.

=> npm install <package-name>:-

- Installs a specific package and adds it to the 'dependencies' section in package.

=> npm uninstall <package-name>:-

- Removes a package from the project and updates the 'package.json' file accordingly.

=> npm search <package name>.

- Searches the npm registry for packages wi the given name.

Q7) Describe use and working of following Node.js Packages. Important properties and methods and relevant programs.

A)

1) Url :-

- The 'url' module provides utilities for URL resolution and parsing. It is used to work with URLs and extract information from them.

eg - Parsing a URL :-

```
const url = require('url');

const urlstring = "some string...";
const parsedUrl = new URL(urlstring);
```

2) Process, pm2 (external package) :-

- 'process' object provides info & control the Node.js process. It allows interacting with the current process & accessing environment variable.

- Getting Command - Line Arguments.

- 'pm2' is an external package used to manage Node.js processes. It provides tools for process monitory, scaling and cluster.

```
npm -i -g pm2.
```

8> readline :-

- The 'readline' module provides an interface for reading input streams line by line. It is commonly used to interact with users in the command-line environment.

- Reading User Input :-

```
const readline = require('readline');
const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout.
});
rl.close();
});
```

4) 'fs' :-

- The 'fs' module provides file system-related functionality allowing reading, writing, and manipulating files.

```
const fs = require('fs');

fs.readFile(string filename, type callback);
```

5) 'events' :-

- The 'events' module provides an event-driven architecture for building appln that can emit and listen to even applns that can emit and listen to events.

```
const EventEmmiter = require('events');

const myEmitter = new myEmitter();

myEmitter.on('greet', (name) => {
    //operation
});

myEmitter.emit('greet', 'Arya');
```

6) console :-

- The console module provides a simple debugging console that can be used to log messages during development.

```
console.log(" ");
console.error(" ");
console.warn(" ");
```

7) Buffer :-

- The 'buffer' module provides a way to handle binary data. It is used to work with raw binary data in Node.js apps.

```
var buf = Buffer.from('abc');
console.log(buf);
```

8) query strings :-

- The query string module provides utilities for working with query strings in url.

9) http:-

- The 'http' module provides a set of functions and classes to create HTTP servers and make HTTP requests.

10) v8:-

- The 'v8' module exposes APIs related to V8 y's engine, providing access to performance and memory related data.

11) os:-

- The os module provides operating system related functionality, such as information about the host operating system.

12) zlib:-

- The 'zlib' module provides compression and decompression functionalities using gzip and deflate.