Backend Syllabus (Node js)

1. Introduction to Backend Development

- Backend development overview
- Difference between frontend and backend
- Client-Server architecture
- libuv library

2. Introduction to Node.js

- What is Node.js?
- Difference between Node.js and browser JavaScript.
- For checking node js version : enter in terminal ‘ node -v ‘
- For checking npm js version : enter in terminal ‘ npm -v ‘
- For manually update npm: ‘ npm install -g npm@latest ‘
- Understanding npm (Node Package Manager)
- Installation and setup of Node.js
- package.json is configuration file : enter in terminal ‘npm init -y’
- package-lock.json : it is automatically install with package.json
- .env file : it is environment variable
- nodemon : it is development tool for automatically restart server after any changes

3. File System (fs module)

- Introduction to File System module
- Importance of synchronous and asynchronous file operations
- **Synchronous (Blocking) ……**
    - Writing Files:
        - Eg. fs.writeFileSync('./test.txt', "hello world");
    - Reading File content:
        - let fileData = fs.readFileSync('./contact.txt', "utf-8");
    - Appending data to file:
        - Eg. fs.appendFileSync('./contact.txt', `\n data`);
    - Copying file data:
        - fs.cpSync('./div.txt', './tempCp.txt');
    - Deleting File

        - fs.unlinkSync('./tempCp.txt');
    - Creating directories
        - fs.mkdirSync('./parentFolder/childFolder', { recursive: true });
    - Removing directories
        - fs.rmSync('mydocs', { recursive: true });
    - Moving file to another directories or rename file
        - fs.renameSync('test.txt', './parentFolder/test.txt');

- **Asynchronous (Non-Blocking) ……..**
    - Writing Files:
        - Eg. fs.writeFile('./test.txt', "hello world", (err) => { if (err) console.log(err); });
    - Reading File content:
        - fs.readFile('./contact.txt', "utf-8", (err, data) => {
          if (err) console.log(err);
          else console.log(data);
          });
    - Appending data to file:
        - fs.appendFile('./contact.txt', `\nNew Data`, (err) => { if (err) console.log(err); });
    - Copying file data:
        - fs.cp('./div.txt', './tempCp.txt', (err) => { if (err) console.log(err); });
    - Deleting File
        - fs.unlink('./tempCp.txt', (err) => { if (err) console.log(err); });
    - Creating directories
        - fs.mkdir('./parentFolder/childFolder', { recursive: true }, (err) => {
          if (err) console.log(err);
          });
    - Removing directories
        - fs.rm('mydocs', { recursive: true }, (err) => { if (err) console.log(err); });
    - Moving file to another directories or rename file
        - fs.rename('test.txt', './parentFolder/test.txt', (err) => { if (err) console.log(err); });

4. Path Module

- Introduction to Path module
- Common Path Methods:
- **Relative vs Absolute paths**
- Cross-platform path handling

    - path.basename() : return last name of file or dir from path
    - path.dirname(): return the parent directory path of any file or dir path

- path.extname(): return extension of any file
- path.parse(): return object of path includes all data (eg. root, dir, base, name, ext)
- path.format(): it is reverse of path.parse()
- path.join(): it return a normalise and complete path after joining various path segment
- path.resolve() : it return the absolute path of any file path (absolute means complete dir path)
- path.relative(from, to) : it return relative path of any file (relative means desire dir path)

- Windows: C:\users\file.txt (backslashes \).
- Unix/Linux/macOS: /users/file.txt (forward slashes /).
- path.sep : use to see our slashes
- __dirname: show absolute path of current dir
- __filename: show absolute path of current file

5. Creating Your Own Module

- Introduction to Modules in Node.js
- module.exports and require basics
- Exporting single and multiple functions/objects
- Understanding module scopes

6. Express.js Basics

- Introduction to Express.js
- Installing and setting up Express
- Creating a basic server with Express
- Understanding routes and route handling
- Middleware functions in Express

7. Advanced Express.js

- Creating **RESTful APIs**
- Understanding HTTP methods (GET, POST, PUT, DELETE)
- Structuring an Express app with **routes and controllers**
- Error handling and creating custom middleware
- Using third-party middleware

8. Introduction to MongoDB

- What is MongoDB?
- Installing and setting up MongoDB locally and globally
- Understanding **collections and documents**
- Basic CRUD operations in MongoDB (Create, Read, Update, Delete)

9. Integrating MongoDB with Node.js (Using Mongoose)

- Introduction to Mongoose
- **Setting up Mongoose and connecting to MongoDB**
- Defining schemas and models
- Performing CRUD operations with Mongoose
- Validation and schema options in Mongoose

10. Building a Full CRUD Application

- Designing the structure of a simple CRUD application
- Creating **models, routes, and controllers**
- **API Development**
- API versioning and documentation // by app.use('/api/v1', router)
- Testing APIs using **Postman**
- form Handling, submissions, and data storage
- Implementing basic validation and error handling
- const cors = require('cors');
- Cross-Origin Resource Sharing (CORS) setup and security
- File uploading
- const fileUpload = require('express-fileupload');
- app.use(express.json())

Set-up:

```
app.use(cors({
   origin:"*"
}))

app.use(fileUpload({
   useTempFiles: true,
   tempFileDir: '/tmp/'
}));
```

11. Cloudinary Setup

- const cloudinary = require("cloudinary").v2;
- connection set up: -

```
cloudinary.config({
    cloud_name: data by cloudinary profile,
    api_key:  data by cloudinary profile,
    api_secret:  data by cloudinary profile,
  });
```

- Media uploading function: -

```
const options = {folder,resource_type:'auto'}      //folder is a name of cloudinary profile folder
const uploadResult = await cloudinary.uploader.upload(file.tempFilePath, options)
```

12. Email Sending

- Setting up email functionality using libraries like **nodemailer**.
- Find App password of Gmail

```
const nodemailer = require("nodemailer");

const  transporter = nodemailer.createTransport({
    host:   smtp.gmail.com,
    auth:{
      user:  admin@gmail.com,
      pass:  App Password
    }
  })


const mailResponse = await transporter.sendMail({
    from:
    to:
    html:
    subject:
  })
```

13. Authentication and Authorization

- **User Authentication**
    - Password hashing with **bcrypt**.
    - Session-based authentication.
    - Using cookies
    - Token-based authentication (**JWT**).
    - Generating JSON Web Tokens.
    - Verifying and decoding tokens for authentication
    - Storing tokens securely (e.g., HTTP-only cookies or local storage).
- **User Authorization**
    - Role-based access control.
    - **Creating Middleware for protecting routes.**
    - Using middleware to verify user roles and permissions.