Backend Development

Class 1
=======
Backend Development

HTTP is a protocol for client and server communication.
4 Types of request in HTTP protocol
-GET - Data Retrieve/Fetch
-PUT - Data Update
-POST - New Data Create/Submit (insert)
-DELETE - Remove Data

What is Idempotent => What all requests are idempotent from above?
ANS - API requests that ensures repeating the operation multiple times produces the same result as executing it once
Idempotent == Uniqueness

Creating New Server Using Express
==========
Express is framework which used to create a server side application

To RUN server
> node server.js (filename)

POST -
It is used to create a record
It is not Idempotent because we hit same request multiple times(with same data) then It will create different records for each request
This case is handled by developers in code so that new records should not get added for same entry

PUT
It is used to update the record
It is Idempotent - Even If we hit a request multiple times, It will update the same record of that request and won't create new entries, uniqueness is there!

We are using Postman, to test the API
Postman is used for design, testing and documentation of API

We are getting error in API, because we are not able parse the data - SO we need to use bodyParser
const bodyParser = require("body-parser");

app.use(bodyParser.json());

We need body-parser to parse/fetch/retrieve data from request body

We are sending a post request through the postman to our server and the response sent from the server can be seen in the postman if the request is sent successfully.

MongoDB

No SQL Database
It is open source database
Data is stored in the form of documents, key-value pair, graphs, etc
No-SQL database is known for scalability and performance
Setup -> i)Install MongoDB ii)Mongo Shell - We can execute CRUD command on this
C - Create, R- Read, U- Update, D- Delete

OR
We can directly use MongoDB Compass, It is GUI where we can perform all the CRUD operations

Download - MongoDB Community Server, MongoDB Compass
Perform all the CRUD operations on MongoDB Compass

How to link Express Server to MongoDB?
==========
Optimize Way - Mongoose - MiddleLayer - Also known as ODM Library
Node JS - Data is treated in object form
MongoDB - Data is treated as Document form

Mongoose acts as ODM Library between Express and MongoDB
Mongoose is a Object Data Modeling (ODM) library for MongoDB distributed as an npm package.

Setup

>npm i mongoose

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost:27017/myDatabase', {
    useNewUrlParser : true,
    useUnifiedTopology : true
})
.then(() => {console.log("Connection Successful!")})
```

.catch((err) => {console.log("Received an Error!")})

myDatabase - If we pass the database that is not created then It will create the database with that name

Class 2
======
All logic is defined in server.js or index.js file
Folders
Routes - define routes of various get, put, post - In routes, We'll define To which controller the paths are mapped
Each route is mapped to Controller ie some business logic
Controller means a file in which business logic is defined
Models - Where we can define description or structure of the schema (The data that we want to store in db, we have created a structure of it)
If we want to create a schema related to database then we'll create that in modal
Schema - Description or Structure of your data
Cofig - folder contains a configuration file that we need in project
.env - We can define which port number and url which we are going to use

Controller - Logic
Modal - Structure

NodeMon

We need to restart the server manually every time we make any changes. so we can use Nodemon
It automatically restarts the server whenever there are any code changes

To start the server using nodemon

```
 "scripts": {
    "start": "node index.js",
    "dev": "nodemon index.js"
  },
```

npm run dev - nodemon start
npm run start - normal start

Define URL and PORT in .env

How are we able to access environment variables anywhere in the project using process.env?
We need to feed the env variables in process object

For that we need to use dotenv library
> npm i dotenv

#require("dotenv").config() - By adding this line in the any file of project, we can use variable defined in .env

```
import dotenv from 'dotenv';
dotenv.config();
```

After writing above lines, whatever we have written in env file will load in process object and we will be able to access it in that file

Developing Todo Application
—----------
> npm i -y
> npm i express
> npm i nodemon -> add scripts in package.json
> npm i mongoose - for establishing connection with db
>npm i dotenv

1. Create database.js in config folder, we need to write config to connect to database
2. Create Todo.js in model to create structure of data

Class 4
—----------
Npm install express
Step 2 :
1. Install npi init -y

Class 5
—----------
What is **Middleware**?
- The request we get, we can parse it, log it, authenticate it, error handling
- To add the middleware in app, you can use app.use()
- app.use(express.jaon())

What is **Mounting**?
- Appending a paths on some other path like /api/v1 and appending it to /getPosts

ODM - Object Data Modeling
It is a library which is used for interacting with noSQL databases mainly. In node js or express application, the data is in the form of an object and in noSql database, the data is stored in the

form of documents. ODM library creates an abstraction layer to map the data in Object and Document form.

MVC - Model-View-Controller

Optimal folder structure
Schema in Class Form -> Model ->
When we create a object of model, it is called document
Multiple documents are called collections

Authentication and Authorization
—----------
Authentication - Identification or verification of registered user
Authorization - Checking for Access rights or Permission

JWT Tokens and Cookies are used to achieve authentication and authorization

JWT Tokens
—----------
JSON Web Tokens
It consists of header, payload and signature . Client sends username and password to server, server creates JWT token and sends it to client. Clients can use the same token to further authentication.

Cookie
—-------
A small text file stored on a client computer which is sent by client. Cookies are used to store user preferences, shopping cart data, user authentication information.

Middlewares
—----------
Middlewares are used to create protected routes
/student route will be access by user who has role student
/admin route will be access by user who has admin access