JSON Web Tokens (JWT)

A JSON Web Token (JWT) is a JSON object which is used to securely transfer information over the web (between two parties). It can be used for an authentication system and can also be used for information exchange.

It is composed of three parts:

- A header
- A payload
- A signature

A header typically contains the type of the token, and the signing algorithm used.

The payload contains the data being transmitted, such as the user's ID or email address.

The signature is created by hashing the header and payload using a secret key, which can be used to verify the authenticity of the token.

Cookies

Cookies in simpler terms means just the textual information about some website.

They are commonly used to store user preferences, shopping cart items and session data. Cookies can also be used for authentication and authorization.

Cookies can be either session cookies or persistent cookies. Session cookies are stored in memory and are deleted when the user closes their browser.

Persistent cookies are stored on the user's computer and are not deleted when the user closes their browser.

Authentication

Authentication is a term that refers to the process of proving the identity of the computer system is genuine.

Authorization

Authorization is the funcation of specifying access right to resources, which is related to general information security and computer security. and to access control in particular.

NOTE: Mostly, above concepts are based on implementation.

Back-end Development

Express.js

It is a small framework that works on the top of Node.js web server functionality to simplify its API and add helpful new features.

It makes it easier to organize your application's functionality with middleware and routing.

* Create a Application

Step 1: create a folder

Step 2: open cmd and move into that folder

Step 3: npm init -y

Step 4: open that folder using VScode

Step 5: npm i express

Step 6: create index.js

* Code

```
const express = require("express");
const app = express();
```

Express framework is imported

Backend application named "app" is created.

```
const port = 3000;
```

callback function

```
app.listen(port, () => {
        console.log('Hello ji')
    })
```

It will listen any communication happens on the given port no.

NOTE: To run this type "node index.js" on terminal.

* Middleware

Middleware are functions that can be used to perform a variety of tasks such as

- Parsing
- Authentication
- Logging
- Error handling

Adding middleware

```
app.use(express.json());
// Its work is to parse/fetch the json.
```

* Routes

Routes handle user navigation to various URLs throughout your application.

HTTP Requests

- GET → Fetch
- POST → Create
- PUT → Update
- DELETE → Remove

Code

```
app.method ("/" (req, res) => {
        res.send (`<h1> Hello ji </H1>`)
})
```

// Whenever you go to "/" route the behaviour we want is written on (req, res => {} function.

// To test these requests use POSTMAN.

* Mounting

In Express.js, "mounting" refers to the process of attaching middleware or sub-applications to specific paths in the Express application.

This allows us to create a modular and flexible application structure.

Code

```
const userRoute = require('./routes/user')
app.use('/users', userRoute)
```

What is Mongoose ?

Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js that provides a higher-level abstraction layer for working with MongoDB.

It simplifies the interaction between MongoDB and Node.js.

What is Nodemon ?

It is a command line tool that monitors your Node.js app and automatically restarts your node app when it detects any changes.

You have to add dependencies to run this.

Go to package.json file.

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

Command to install

```
npm i nodemon
```

Command to start Nodemon

```
npm run dev
```

What is MVC ?

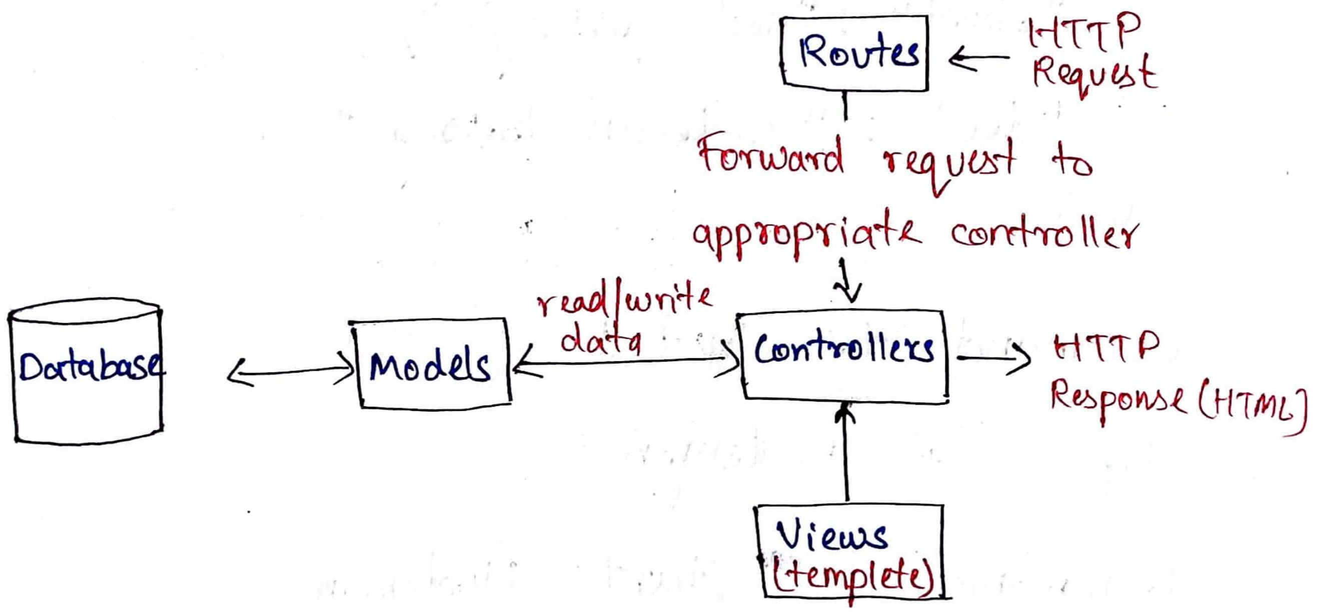
MVC stands for Model-View-Controller, which is a software architectural pattern used in web development to seperate the application's concern into three interconnected components.

Model - It includes schema.

View - It includes frontend templete

Controller - It includes the logic of the code.

* Optimal Folder Structure



* How to Export?

Syntax

```
module.exports = myObject;
```

* Schema

A schema is a blueprint that defines the structure of a document in mongoDB. It describes the fields or properties that a document can have, along with their data types, default values and validation rule.

* Models

A model is a representation of a MongoDB collection. It is created from a schema and provides an interface with the data in the collection. Models used to perform CRUD operations.

* Documents

A document is an instance of a model and represents a single record in a MongoDB collection. It contains fields with values that correspond to the properties defined in the schema.

* Collection

A collection is a group of related documents in MongoDB. It is similar to a table in a relational database and can be queried and maniputated using MongoDB commands.

What is MongoDB ?

It is a NoSQL document - oriented database that is often used in MERN stack applications.

It provides a flexible and scalable database solution that can handle large amounts data.

It stores data in JSON - like documents.