Week 2 Exploration (Mentor - Abinaya)

- **Zainab Nusaiba**

Objective:

1. HTTP Requests
2. Token and its types
3. How to pass tokens in the front-end
4. Create a node application with the following
    - Hello World
    - API call and Routing
    - API with CRUD operations
5. Cookies

1. HTTP Requests

API developers typically use GET, PUT, or POST, but officially there exist 39 total HTTP verbs each providing a method for powerful interactions. . The most commonly used HTTP request methods are GET, POST, PUT, PATCH, DELETE which are equal to the CRUD (create, read, update, and delete) operations.

1. **GET -** to retrieve data from the server using a given URI. On success, it returns an HTTP status code of 200 (OK)
2. **POST -** to send data (file, form data, etc.) to the server and on success, it returns an HTTP status code of 201
3. **PUT -** to modify the data on the server. Replaces the entire data with the passed data. If the request doesn't match, it creates one.
4. **PATCH -** to partially modify a targeted resource
5. **HEAD -** same as GET, but it transfers the status line and the header section only
6. **DELETE -** to remove all the resources from the given URI
7. **CONNECT -** to establish a tunnel with the given URI
8. **OPTIONS -** to describe communication options for the target resource
9. **TRACE -** to perform a message loop-back test along with the path to the target resource.

2. Tokens and their types

- Tokens in C language

Tokens are the smallest elements in a program. The individual units representing tokens and making up a program are meaningful to the compiler. There exist 6 types of tokens in C language and they are

- Keywords
- Identifiers
- Constants
- Strings
- Operators
- Special Characters

- Tokens in node.js

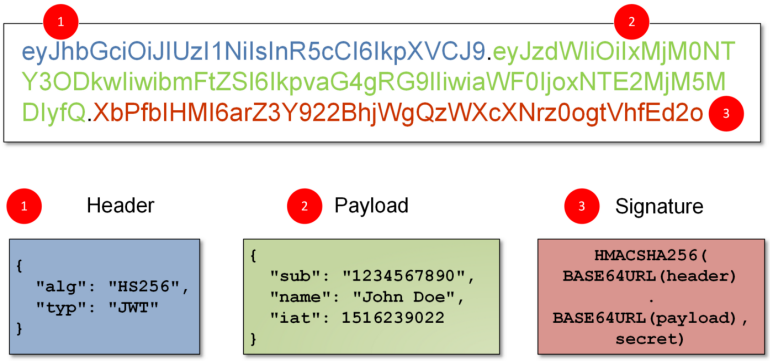
The Tokens of node.js are called JSON Web Tokens (JWT) which is an open standard for secure data interchange using **JSON object**. JWT is used for stateless authentication mechanisms for users and providers.

Stateless Authentication is a way to verify users by having much of the session information such as user properties stored on the client side. It makes identity verification with the backend seamless. Also called token-based authentication.

The user receives a JWT after a successful login, which contains all important information about the user. This means that the session no longer has to be saved on the server and is therefore also called a **stateless session**.

A JWT needs to be stored in a safe place inside the user's browser. Anyway, you shouldn't store a JWT in local storage (or session storage). If you store it in a LocalStorage/SessionStorage then it can be easily grabbed by an XSS attack.

The Figure shows that a JWT consists of three parts: a header, payload, and signature.



3. Create a node application

- Hello World

A simple file created with the help of node.js application that **console.log** aka prints the message

- API calls

The API Calls done with the crud app deals with **RESTful API.** The RESTful API communicates with the HTTP requests found in browsers and servers. This is two-way communication. Some of the HTTP methods that are present in the RESTful API include PUT, GET, PATCH, DELETE, and UPDATE.

- Routing

Routing refers to the application's **endpoints** referring to the **URIs** that are responding to the client's requests. Routing is defined using the basic HTTP methods which have two parameters and the first one directs to the endpoint or the routing. The program I have executed involves express. Hence the routing is done with the help of an express app object along with the RESTful API calls with the help of HTTP methods.

Eg: **app. get()** handles the GET request

```
app.get("/", (req, res) => {
        db.collection("quotes")
```

```
        .find()
        .toArray()
        .then((results) => {
        console.log(results);
            res.render("index.ejs", { quotes: results });
        })
        .catch((error) => console.error(error));
});
```

Eg: **app. post()** handles the POST request

```
app.post("/quotes", (req, res) => {
  quotesCollection
    .insertOne(req.body)
    .then((result) => {
      console.log(result);
      res.redirect("/");
      console.log(req.body);
    })
    .catch((error) => console.error(error));
});
```

- CRUD operations

CRUD apps are the user interface that we use to interact with databases through APIs. It is a specific type of application that supports the four basic operations: Create, read, update, and delete.

The Concepts used in this include API, routing, database connection, and storing. All of these are connected to the basic HTTP requests. Therefore Broadly, CRUD apps consist of the database, the user interface, and the APIs.

The technologies used to build this CRUD app are

- ES6
- Express
- Javascript
- CSS
- HTML
- MongoDB Atlas

Operation	HTTP Request	Database Method
Create	POST	.inserOne()
Read	GET	.find()
Update	PUT	.findOneAndUpdate()
Delete	DELETE	.deleteOne()