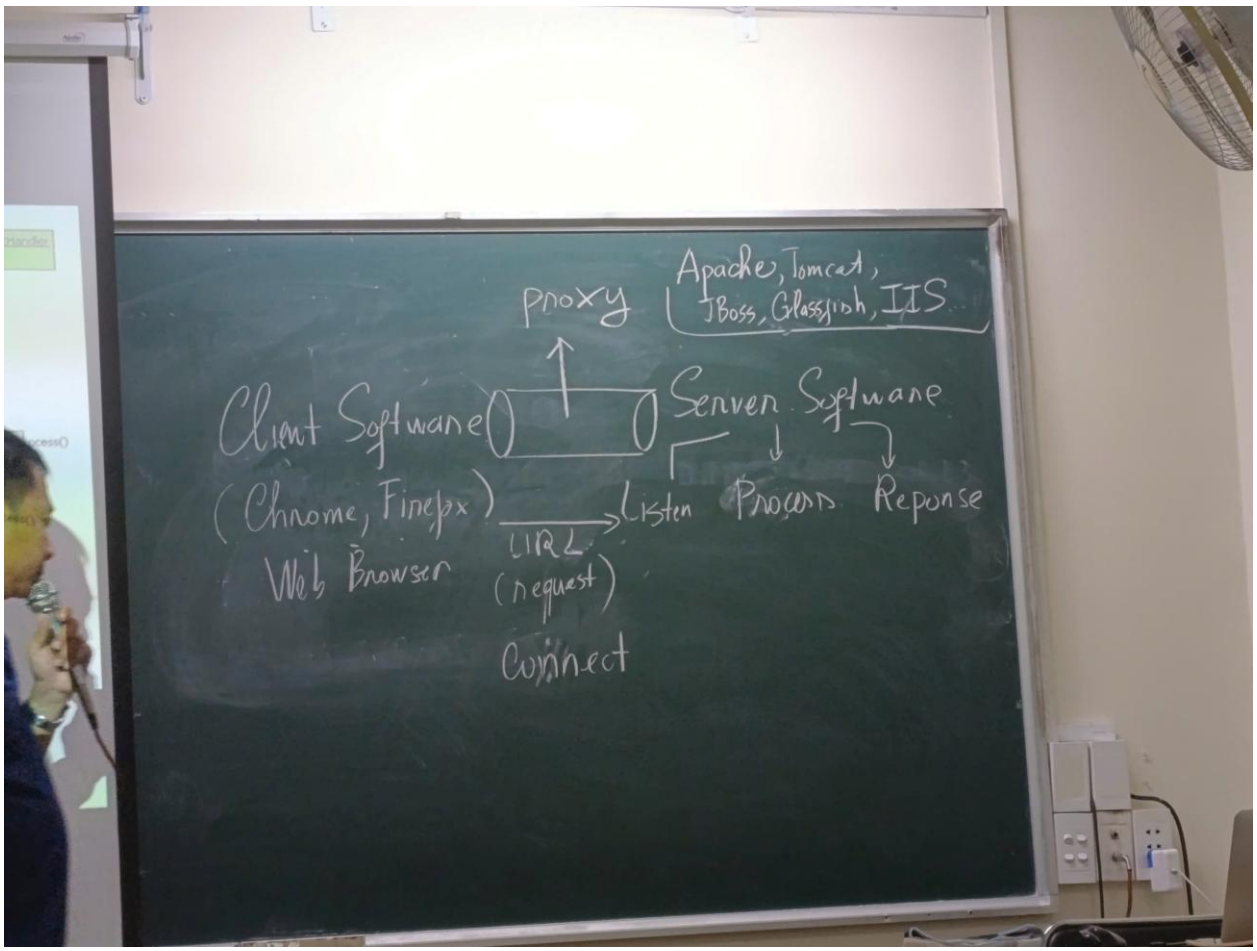
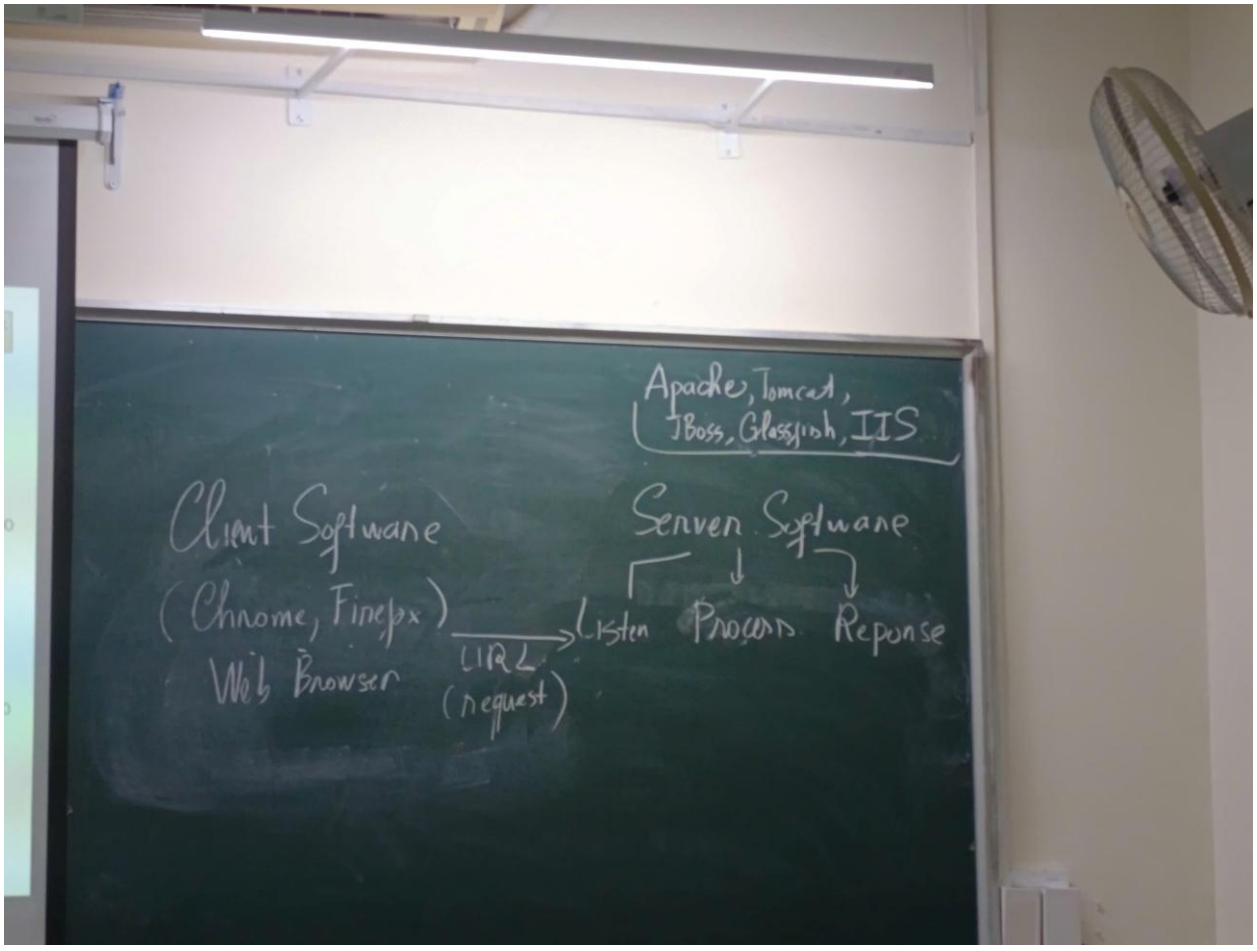
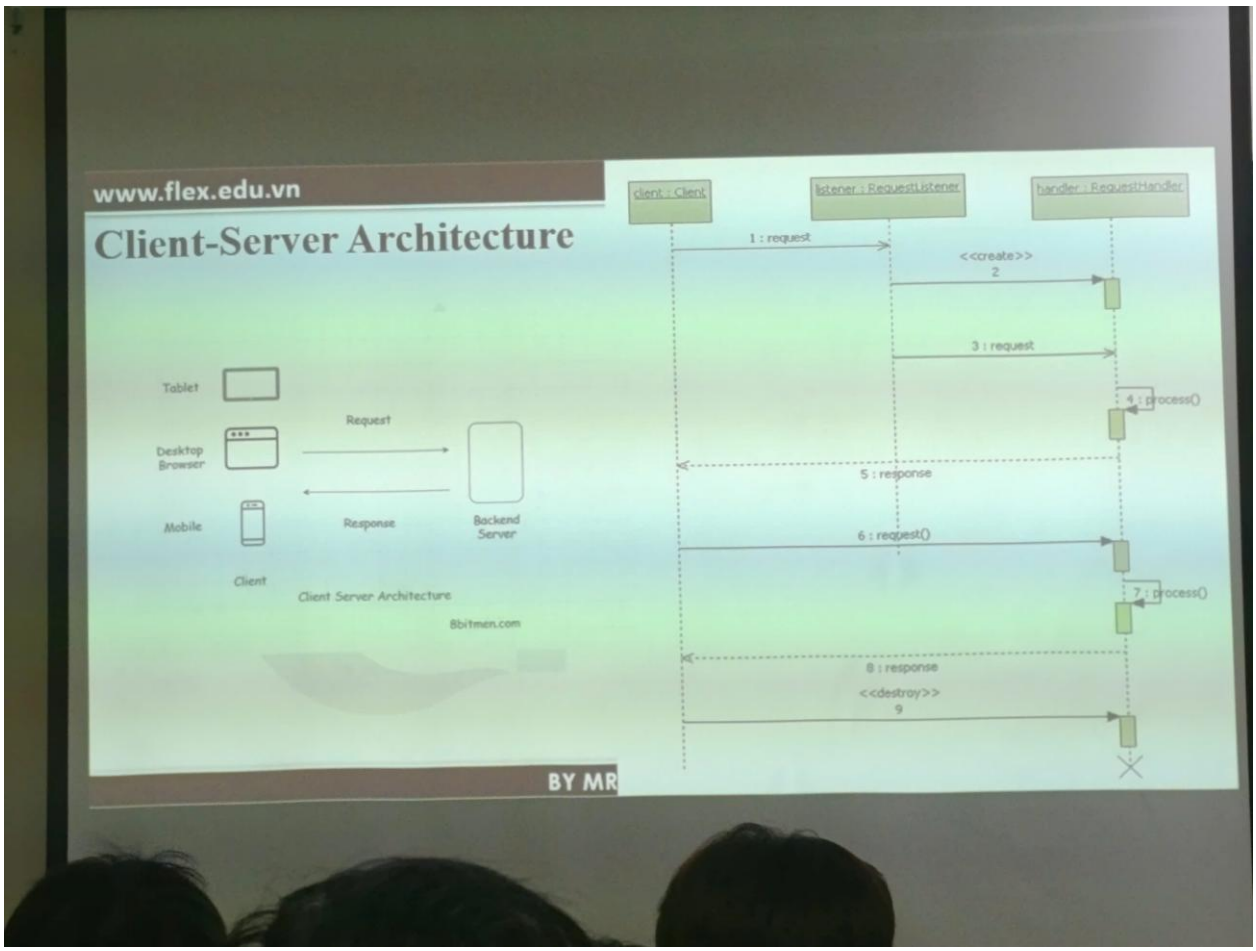
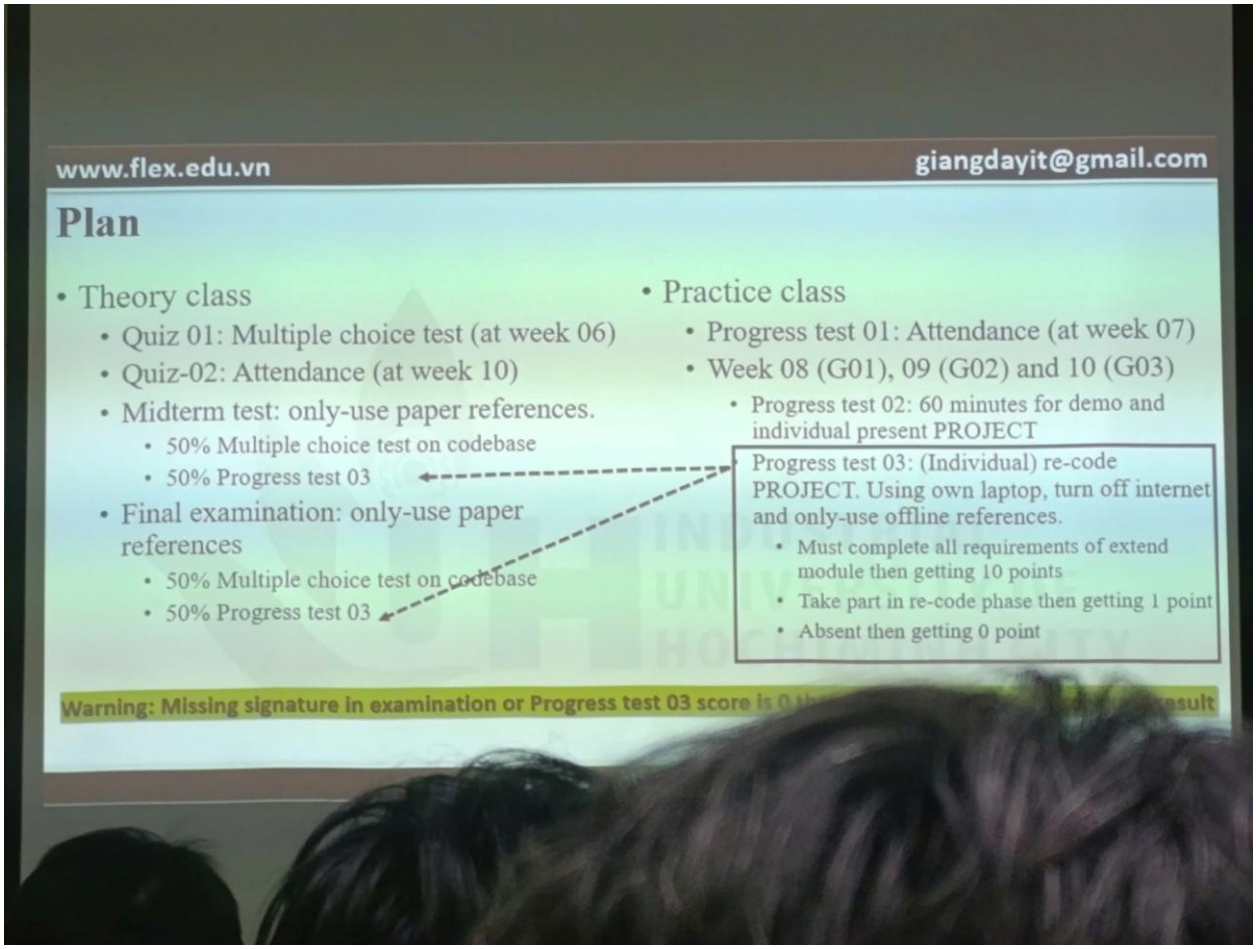

proxy
Apache, Tomcat,
JBoss, Glassfish, IIS
Client Software
(Chrome, Firefox)
Web Browser
URL
(request)
Connect
Server Software
Listen
Process
Reponse

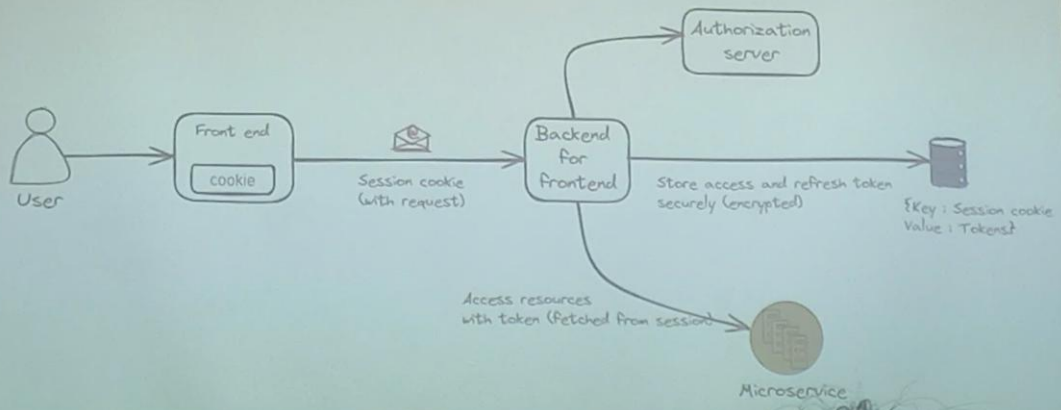

Apache, Tomcat,
JBoss, Glassfish, IIS
Client Software
(Chrome, Firefox)
Web Browser
URL
(request)
Server Software
Listen
Process
Reponse


www.flex.edu.vn
Client-Server Architecture
Tablet
Desktop Browser
Mobile
Client
Request
Response
Backend Server
Client Server Architecture
8bitmen.com
client : Client
listener : RequestListener
handler : RequestHandler
1 : request
<<create>>
2
3 : request
4 : process()
5 : response
6 : request()
7 : process()
8 : response
<<destroy>>
9
BY MR


www.flex.edu.vn
giangdayit@gmail.com
Plan
• Theory class
• Quiz 01: Multiple choice test (at week 06)
• Quiz-02: Attendance (at week 10)
• Midterm test: only-use paper references.
• 50% Multiple choice test on codebase
• 50% Progress test 03
• Final examination: only-use paper references
• 50% Multiple choice test on codebase
• 50% Progress test 03
• Practice class
• Progress test 01: Attendance (at week 07)
• Week 08 (G01), 09 (G02) and 10 (G03)
• Progress test 02: 60 minutes for demo and individual present PROJECT
Progress test 03: (Individual) re-code PROJECT. Using own laptop, turn off internet and only-use offline references.
• Must complete all requirements of extend module then getting 10 points
• Take part in re-code phase then getting 1 point
• Absent then getting 0 point
Warning: Missing signature in examination or Progress test 03 score is 0

Securing session data

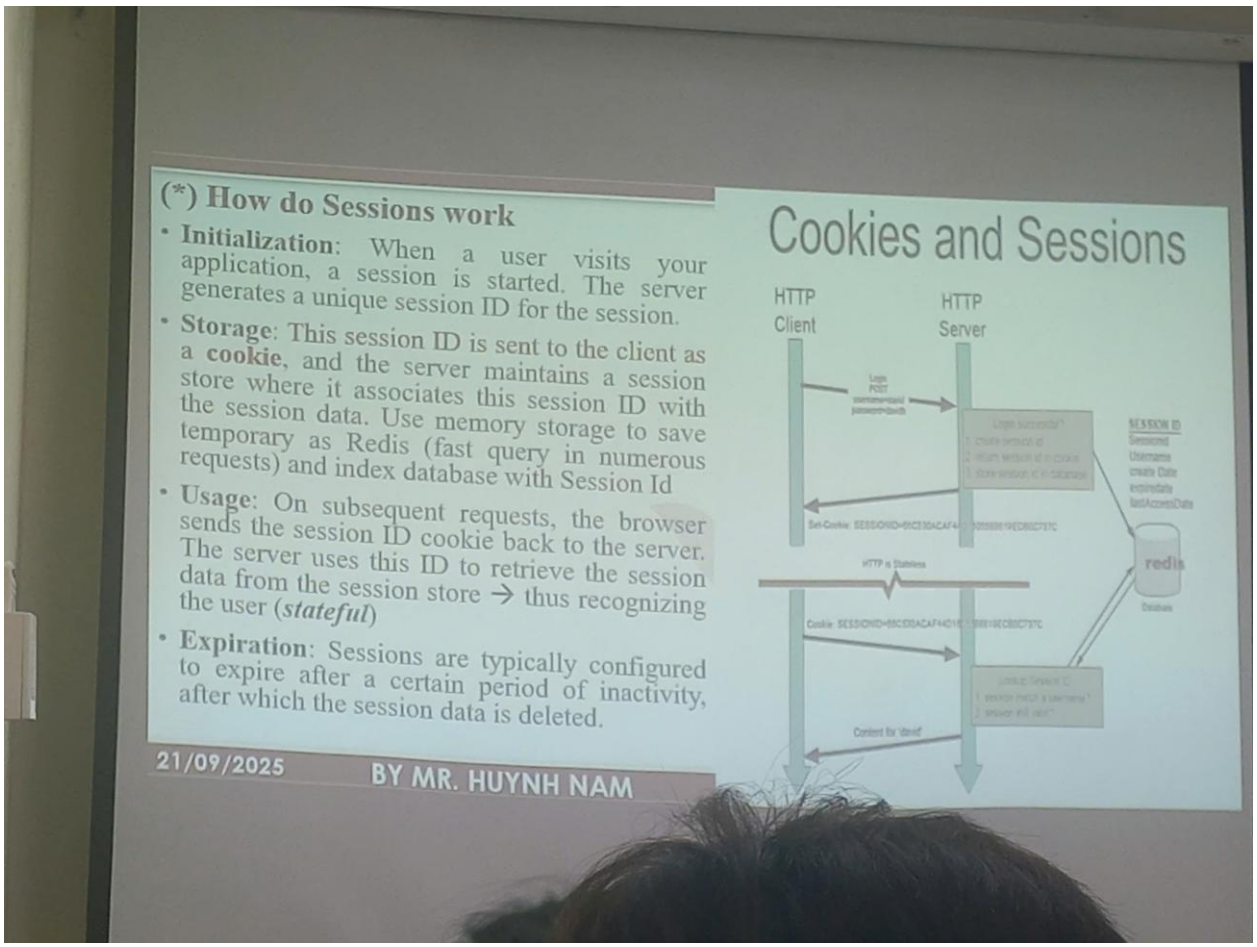
- It is crucial to secure session data to prevent unauthorized access or tampering.
- You can secure session data by using secure cookies, encrypting session data, and implementing HTTPS encryption.

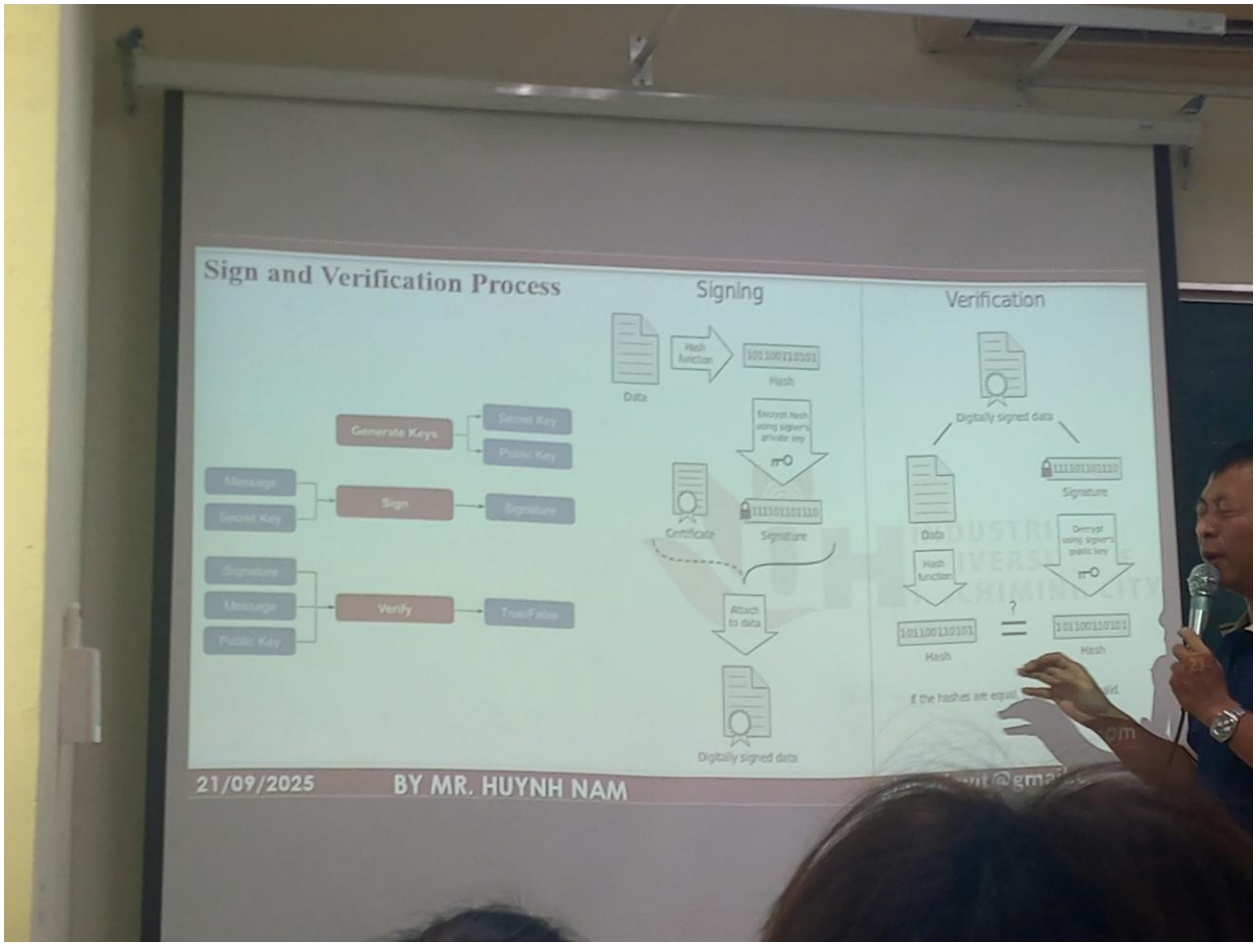


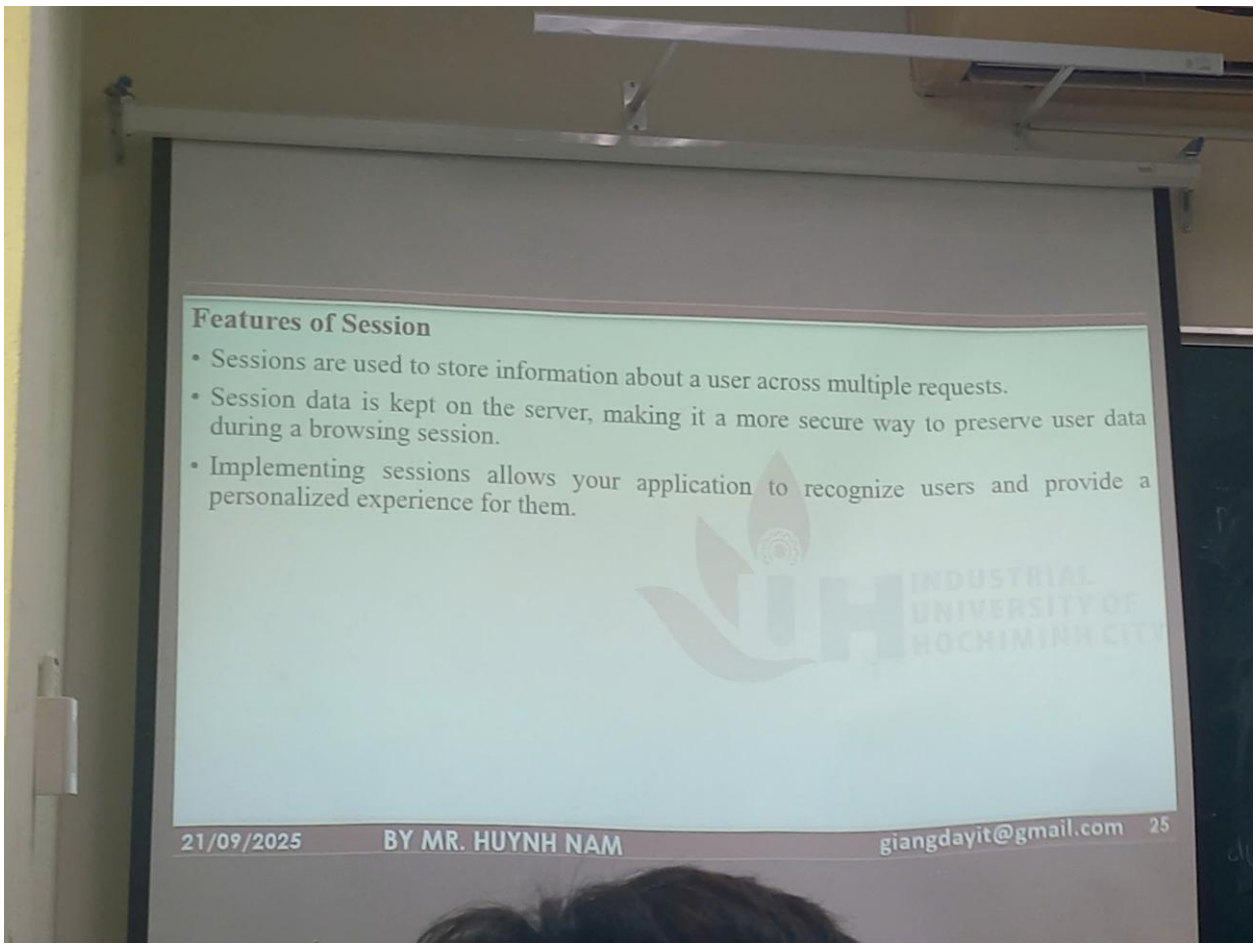
What is Passport

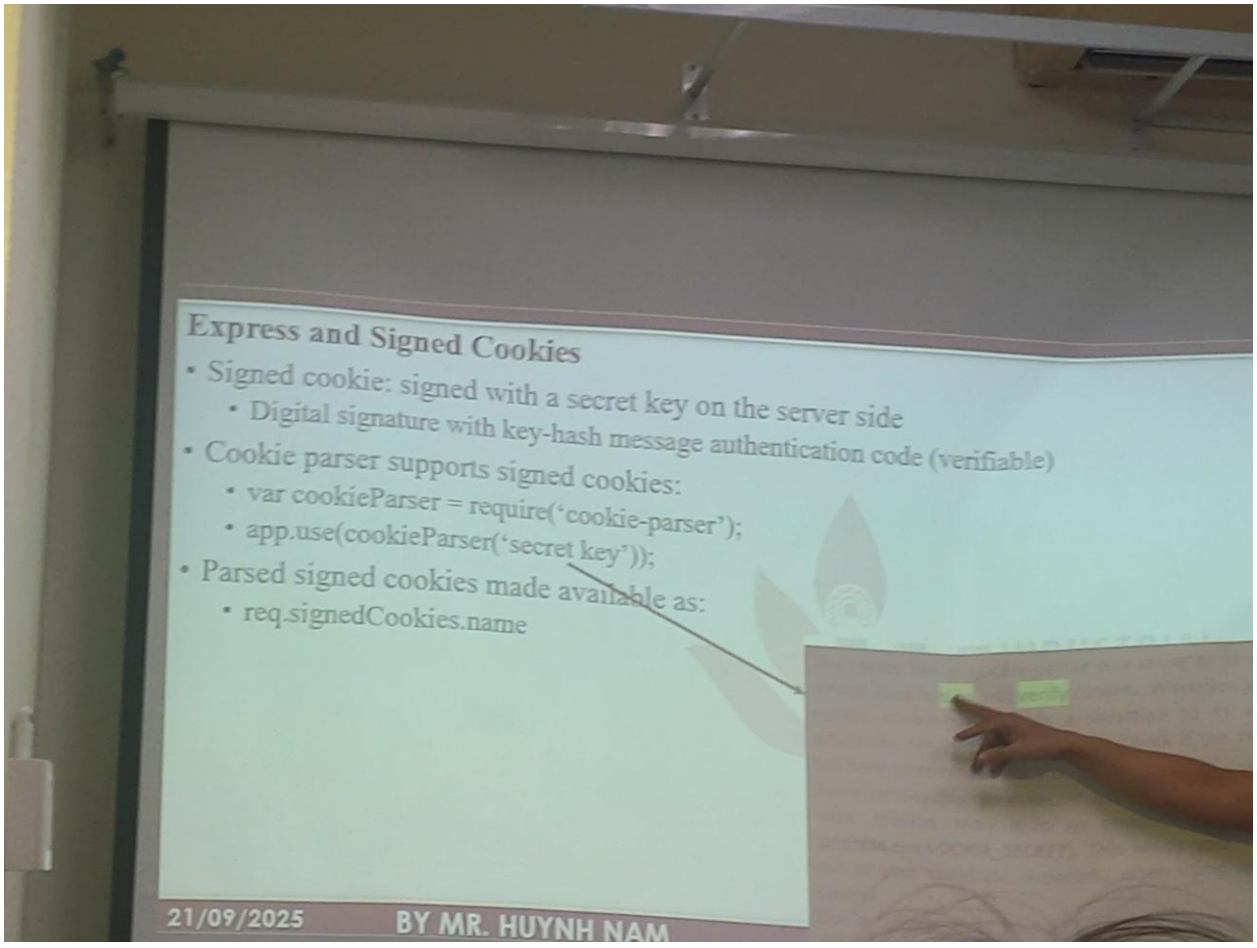
- Passport is Express-compatible authentication middleware for Node.js.
- Passport's sole purpose is to authenticate requests, which it does through an extensible set of plugins known as strategies.
- Passport does not mount routes or assume any particular database schema, which maximizes flexibility and allows application-level decisions to be made by the developer.
- The API is simple: you provide Passport a request to authenticate, and Passport provides hooks for controlling what occurs when authentication succeeds or fails.
- Supports various strategies for authentication:
  - Local strategy
  - OpenID
  - Oauth (Facebook, Twitter, G+ etc.) single sign-on
- Install Passport: ***npm install passport***

21/09/2025 BY MR. HUYNH NAM


(*) How do Sessions work
• Initialization: When a user visits your application, a session is started. The server generates a unique session ID for the session.
• Storage: This session ID is sent to the client as a cookie, and the server maintains a session store where it associates this session ID with the session data. Use memory storage to save temporary as Redis (fast query in numerous requests) and index database with Session Id
• Usage: On subsequent requests, the browser sends the session ID cookie back to the server. The server uses this ID to retrieve the session data from the session store → thus recognizing the user (stateful)
• Expiration: Sessions are typically configured to expire after a certain period of inactivity, after which the session data is deleted.
21/09/2025
BY MR. HUYNH NAM
Cookies and Sessions
HTTP Client
HTTP Server
redis


Sign and Verification Process
Signing
Verification
Generate Keys
Secret Key
Public Key
Message
Secret Key
Sign
Signature
Signature
Message
Public Key
Verify
True/False
Data
Hash function
Hash
Encrypt hash using signer's private key
Certificate
Signature
Attach to data
Digitally signed data
Digitally signed data
Signature
Data
Hash function
Decrypt using signer's public key
Hash
Hash
21/09/2025
BY MR. HUYNH NAM


Features of Session
• Sessions are used to store information about a user across multiple requests.
• Session data is kept on the server, making it a more secure way to preserve user data during a browsing session.
• Implementing sessions allows your application to recognize users and provide a personalized experience for them.
21/09/2025
BY MR. HUYNH NAM
giangdayit@gmail.com
25

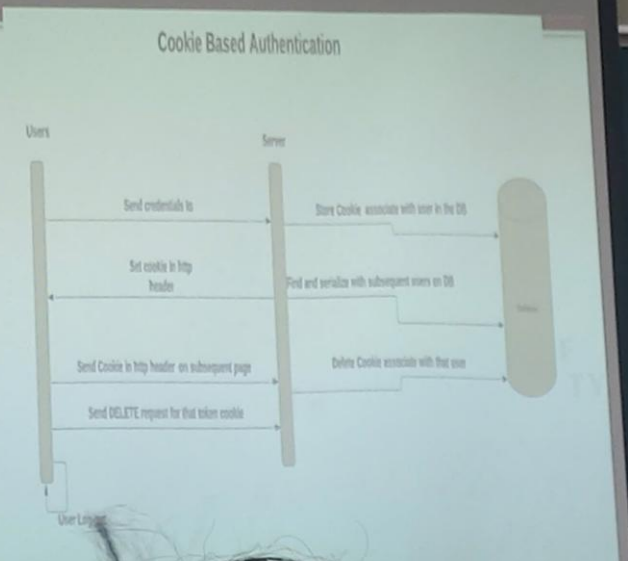

Express and Signed Cookies
• Signed cookie: signed with a secret key on the server side
• Digital signature with key-hash message authentication code (verifiable)
• Cookie parser supports signed cookies:
• var cookieParser = require('cookie-parser');
• app.use(cookieParser('secret key'));
• Parsed signed cookies made available as:
• req.signedCookies.name
21/09/2025
BY MR. HUYNH NAM

Express and Cookies

- Server can set a cookie as follows in any of the middleware:
  - res.cookie(name,value,options)
- Cookies are parsed in Express server using the cookie-parser middleware
  - var cookieParser = require('cookie-parser');
  - app.use(cookieParser());
- Cookie-parser parses incoming cookies and attaches them to request
  - req.cookies.name

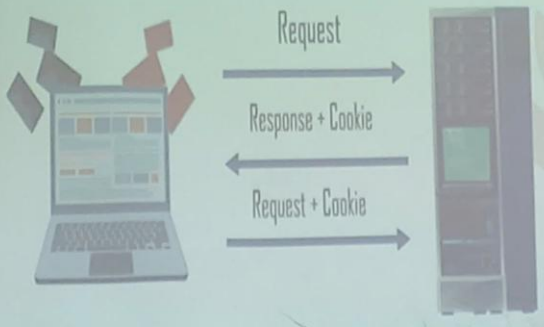
How do Cookies work

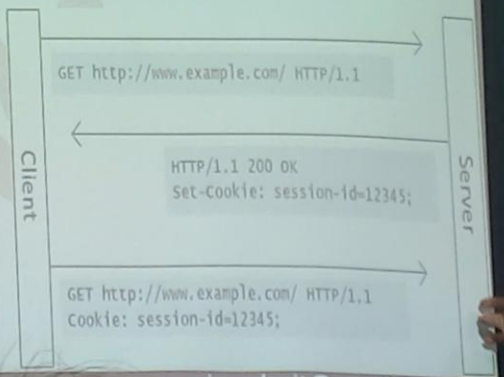
- When working with cookies, it's important to consider security aspects, such as:
  - ***HttpOnly Flag***: Makes the cookie inaccessible to JavaScript's Document.cookie API; it's only sent to the server. This helps mitigate cross-site scripting (XSS) vulnerability attacks.
  - ***Secure Flag***: Ensures the cookie is sent over HTTPS, protecting it from man-in-the-middle attacks.
  - ***SameSite Attribute***: Helps prevent cross-site request forgery (CSRF) attacks. It can be set to Strict, Lax, or None (with the Secure flag).



HTTP Cookies

- Small piece of data **(4KB)** sent from a web server and stored on the client side
- Each subsequent request from the client side should include the cookie in the request **header**

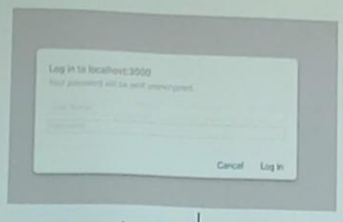




05/03/2024 BY MR. HUYNH NAM giangdayit@gmail.com

Setting up Basic Authentication

- Open the app.js file and update its contents as follows:



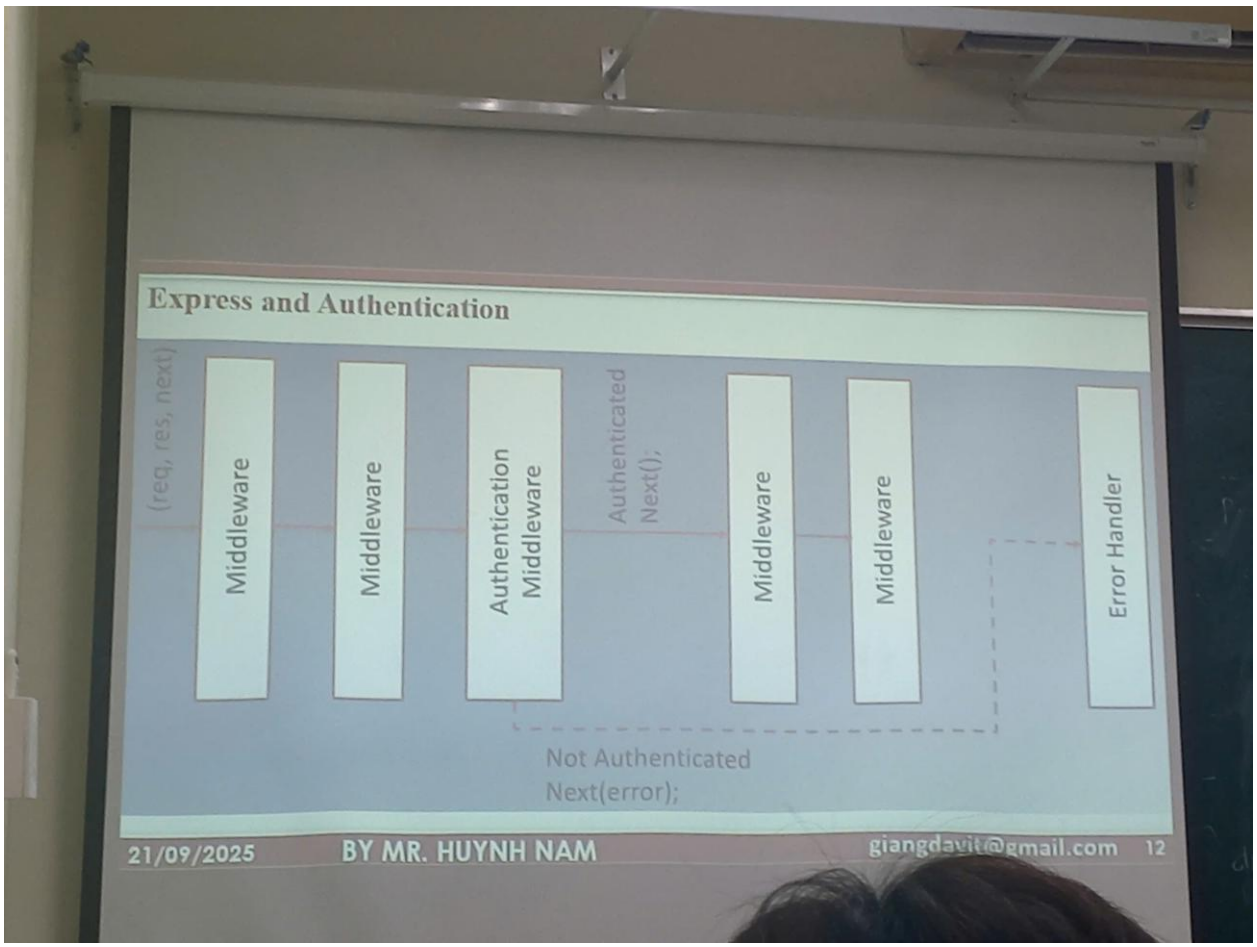
Login Fail

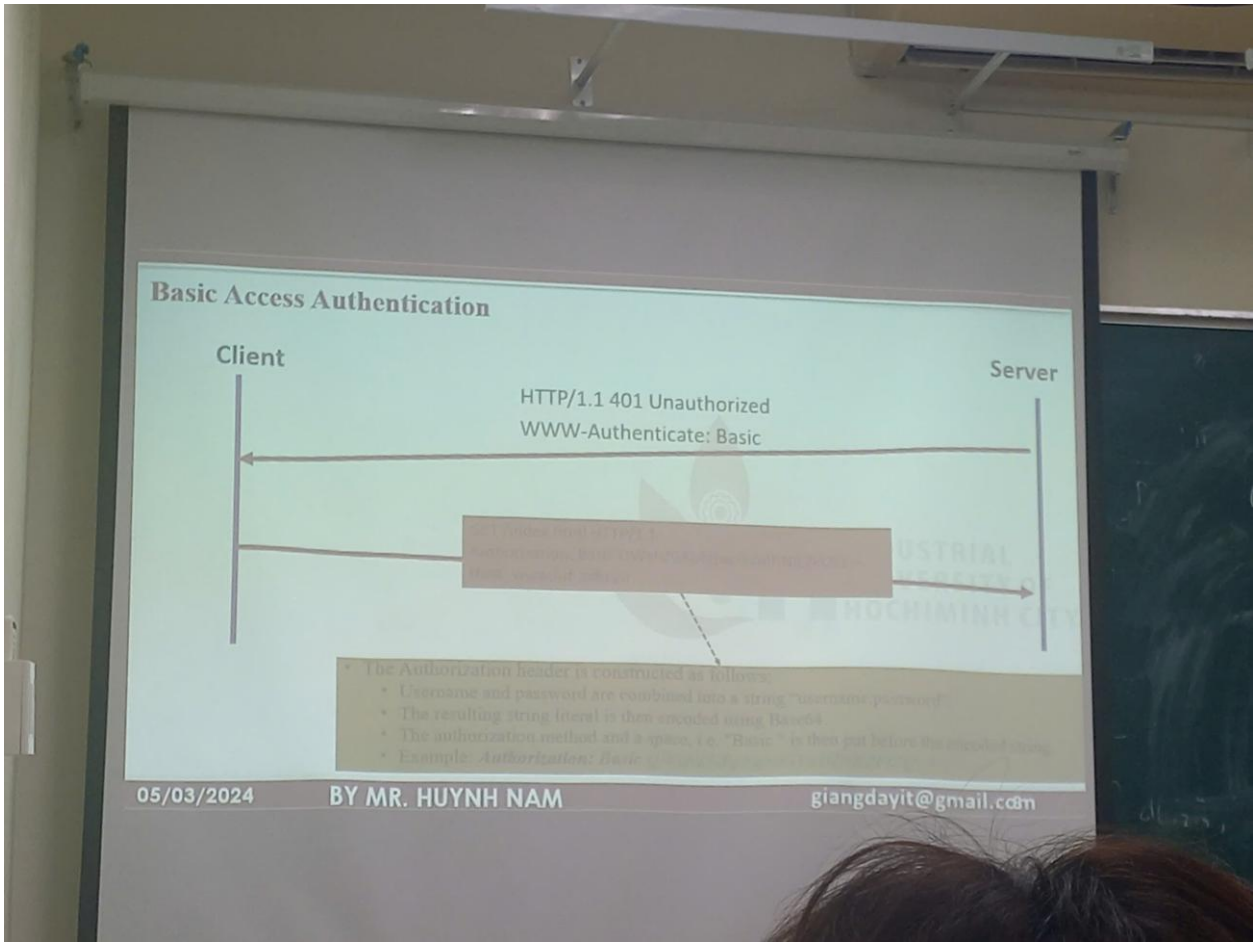
You are not authenticated!

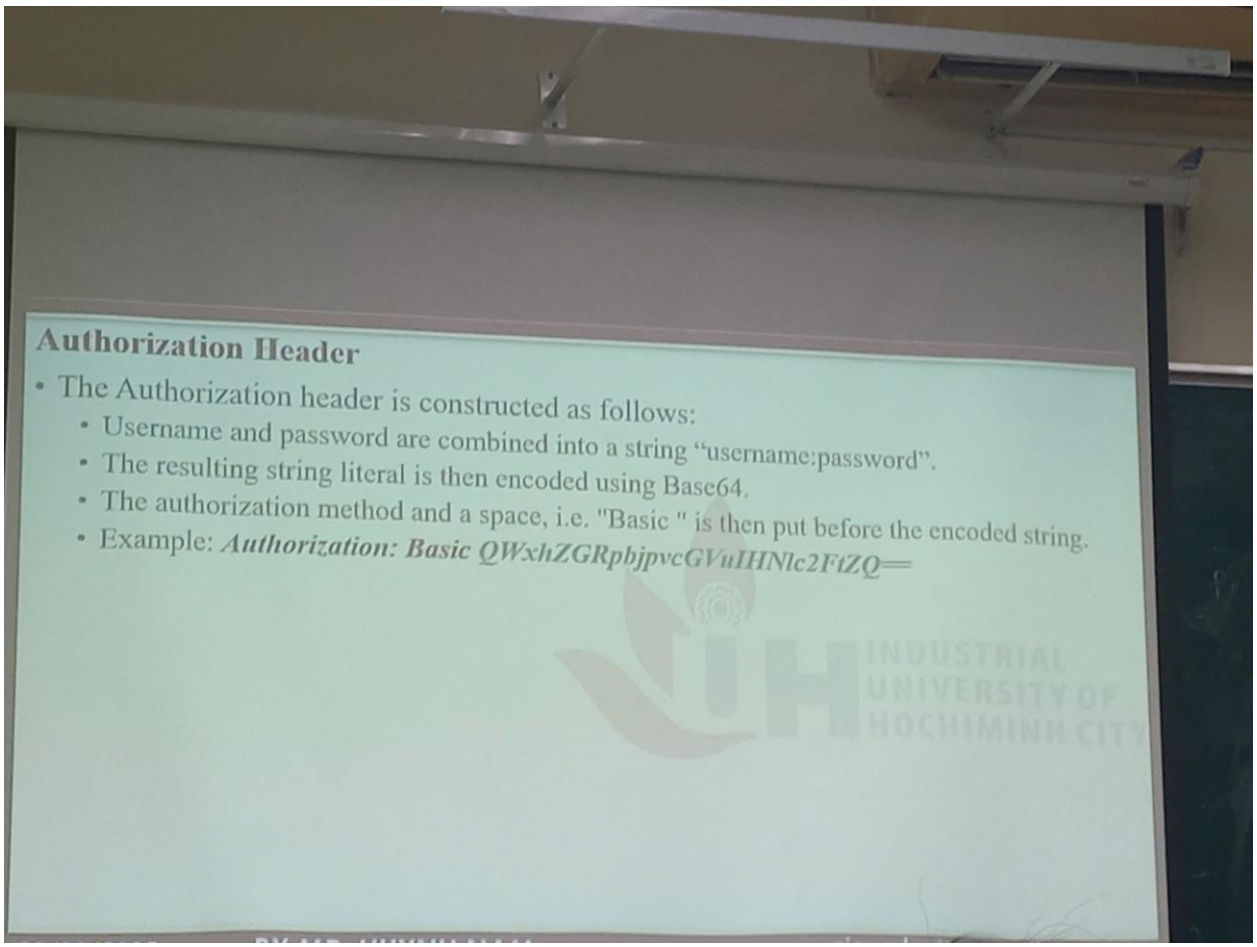
401

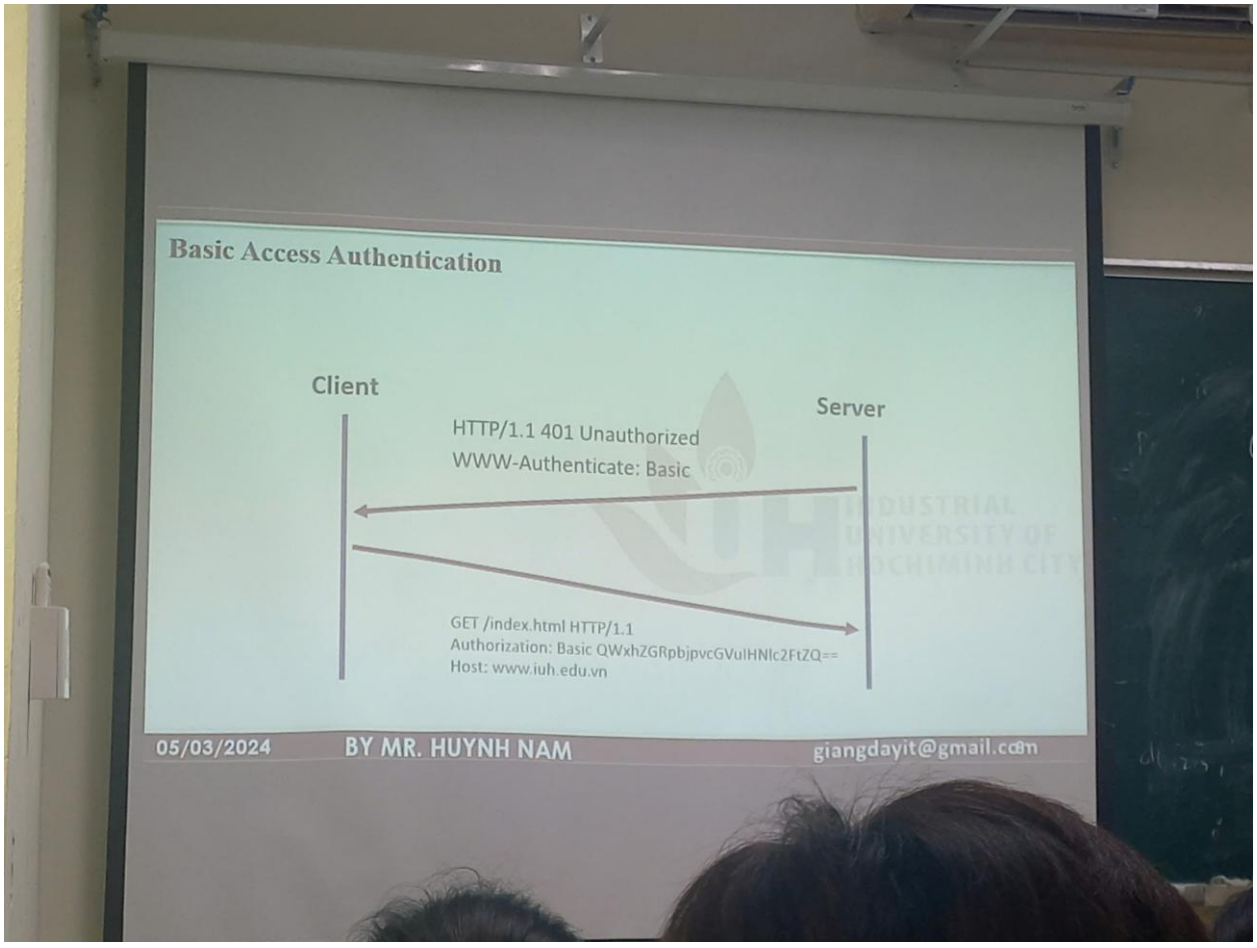
```
function auth (req, res, next) {
  console.log(req.headers);
  var authHeader = req.headers.authorization;
  if (!authHeader) {
      var err = new Error('You are not authenticated!');
      res.setHeader('WWW-Authenticate', 'Basic');
      err.status = 401;
      next(err);
      return;
  }
  var auth = new Buffer.from(authHeader.split(' ')[1],
'base64').toString().split(':');
  var user = auth[0];
  var pass = auth[1];
  if (user == 'admin' && pass == 'password') {
      next(); // authorized
  } else {
      var err = new Error('You are not authenticated!');
      res.setHeader('WWW-Authenticate', 'Basic');
      err.status = 401;
      next(err);
  }
}

app.use(auth);
```

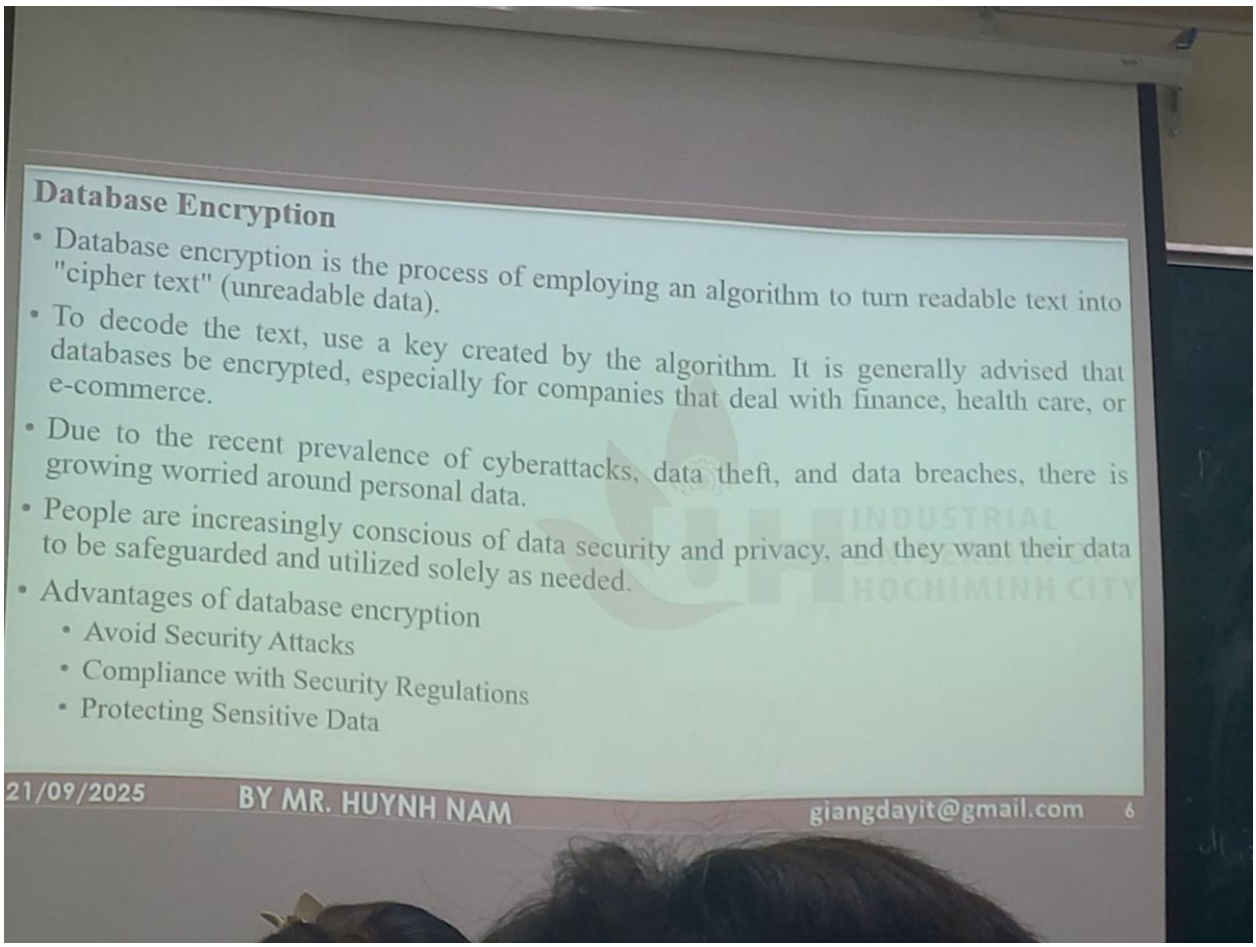
21/09/2025 BY MR. HUYNH NAM

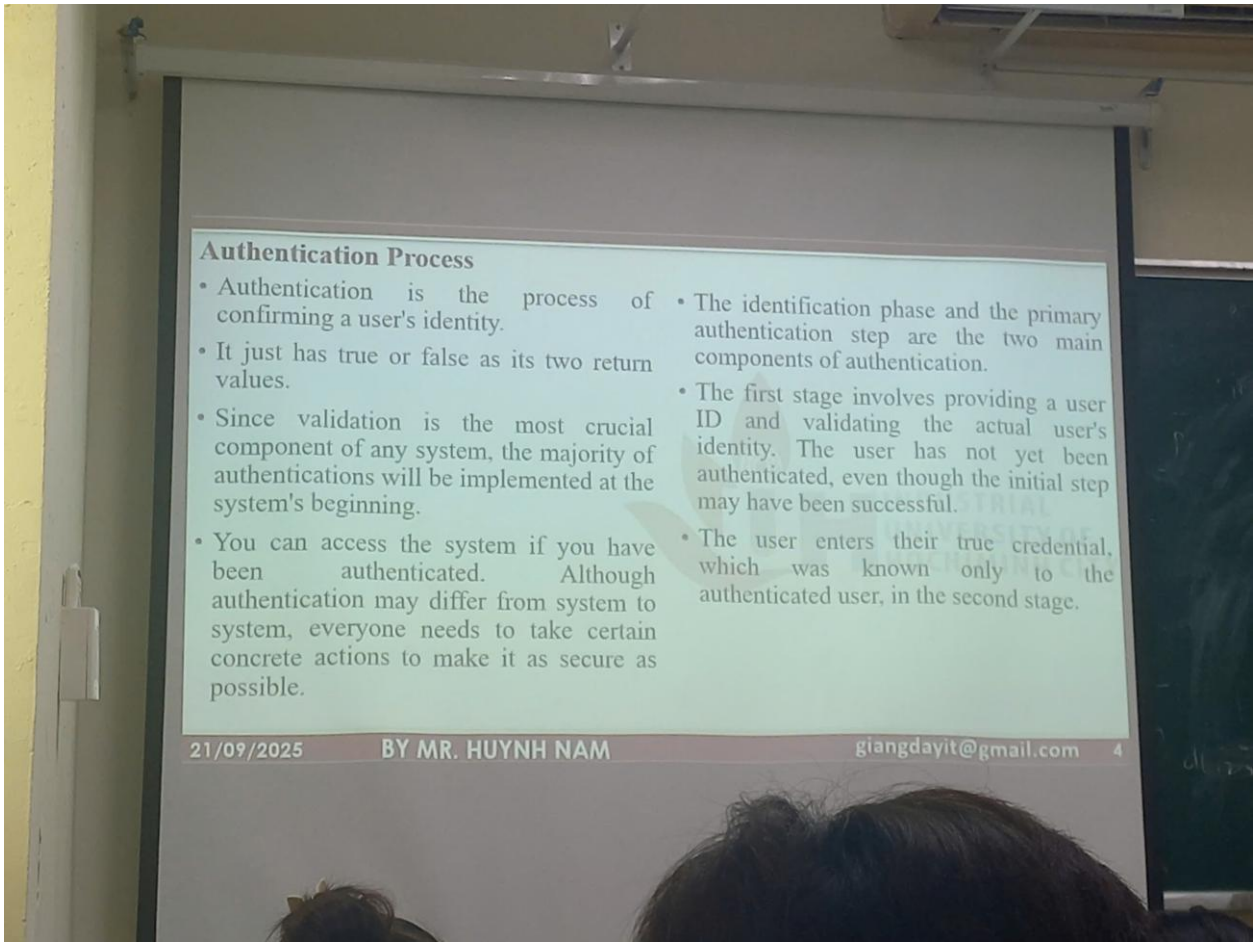

Express and Authentication
(req, res, next)
Middleware
Middleware
Authentication Middleware
Authenticated Next();
Middleware
Middleware
Error Handler
Not Authenticated Next(error);
21/09/2025
BY MR. HUYNH NAM
giangdayit@gmail.com
12


Basic Access Authentication
Client
Server
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Basic
05/03/2024
BY MR. HUYNH NAM
giangdayit@gmail.com


Authorization Header
• The Authorization header is constructed as follows:
• Username and password are combined into a string "username:password".
• The resulting string literal is then encoded using Base64.
• The authorization method and a space, i.e. "Basic " is then put before the encoded string.
• Example: Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==


Basic Access Authentication
Client
Server
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Basic
GET /index.html HTTP/1.1
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
Host: www.iuh.edu.vn
05/03/2024
BY MR. HUYNH NAM
giangdayit@gmail.com


Database Encryption
• Database encryption is the process of employing an algorithm to turn readable text into "cipher text" (unreadable data).
• To decode the text, use a key created by the algorithm. It is generally advised that databases be encrypted, especially for companies that deal with finance, health care, or e-commerce.
• Due to the recent prevalence of cyberattacks, data theft, and data breaches, there is growing worried around personal data.
• People are increasingly conscious of data security and privacy, and they want their data to be safeguarded and utilized solely as needed.
• Advantages of database encryption
• Avoid Security Attacks
• Compliance with Security Regulations
• Protecting Sensitive Data
21/09/2025 BY MR. HUYNH NAM giangdayit@gmail.com 6


Authentication Process
• Authentication is the process of confirming a user's identity.
• It just has true or false as its two return values.
• Since validation is the most crucial component of any system, the majority of authentications will be implemented at the system's beginning.
• You can access the system if you have been authenticated. Although authentication may differ from system to system, everyone needs to take certain concrete actions to make it as secure as possible.
• The identification phase and the primary authentication step are the two main components of authentication.
• The first stage involves providing a user ID and validating the actual user's identity. The user has not yet been authenticated, even though the initial step may have been successful.
• The user enters their true credential, which was known only to the authenticated user, in the second stage.
21/09/2025 BY MR. HUYNH NAM giangdayit@gmail.com 4

List of authentication techniques

- Password authentication
- Multi-factor authentication
- Biometric authentication
  - Facial recognition
  - Fingerprint scanners
  - Voice Identification
  - Certificate-based authentication
  - Token-based authentication

Terms in Security

- Authentication
- Authorization
- User's identity
- Credential

A hook is a way to extend software

- The Unix programming philosophy is to do (just) one thing but do it well. Focusing on one core task is hard as developers see and feel their users' pain points and desires. To be able to focus on the core product but allow additional functions we cannot take care of, we develop plugin systems. One way to do this is by giving hooks.
- Example: Think about an email client. The email client might define a hook *pre_send* that receives the current receiver, subject line, and content of the email. Developers can use this to extend the software. You could have one plugin that uses this hook and adds a signature if it's missing. Another hook could look for swear words and stop you from sending that email drunk at night, which you will regret.
- Another way to extend software is by an event. An event just tells the plugin that something happened.
- You can implement custom "hooks" in your application using patterns like middleware or event emitters to inject logic at specific points in your code.
  - **Middleware:** Functions that execute in sequence before a route handler. They can modify the request/response, perform checks, or add data.
  - **Event Emitters:** Allow you to define custom events and attach listeners (hooks) that execute when those events are emitted.
- ***Callbacks can be used to implement hooks***, particularly in frameworks like React

How to call Call-Back Function

```
function namedCallback(){
   alert("namedCallback()");
}
function testFunction(callback){
   callback();
}
testFunction(namedCallback);
```

```
var name = "MR. HUYNH NAM";

function whoWriteThis(param){
   alert("whoWriteThis() called by "+param);
}

function testFunction(callback){
   callback();
}

testFunction(whoWriteThis(author));
```

```
function performOperation(value, callback) {
  // Do some work with 'value'
  const result = value * 2;
  callback(result); // The calling function passes 'result' to the callback
}

function handleResult(data) {
  console.log("The result is:", data);
}

performOperation(5, handleResult); // Output: The result is: 10
```

```
$("#btn_1").click(function() {
  alert("Btn 1 Clicked");
});

Var someUlr = --;

function successCallback(){
  //success code
}

function completeCallback(){
  //complete code
}

function errorCallback(){
  //error code
}

$.ajax({
  url: someUrl,
  success: successCallback,
  complete: completeCallback,
  error: errorCallback
})
```

Callbacks

- Callbacks are pointers to functions. They represent a concrete function. When you pass a function as a parameter to another function, you call it a callback.
- In JavaScript, functions are objects. Because of this, functions can take functions as arguments and can be returned by other functions. Functions that do this are called higher-order functions. Therefore, a callback function is a function that passes as an argument.
- A callback, as the name suggests, is a function that is to execute after another function has finished executing

A higher order function takes a function as a parameter

```
const higherOrderFunction = (callback) => {
return callback()
}
```

A callback is a function that is passed as an argument

Passing data model from router to EJS

```
//app.js
const express = require('express');
const app = express();
const articles_data = require('./data.json');

app.set('view engine', 'ejs');

app.get('/', (req, res) => {
  res.render('homepage', { articles: articles_data });
});

const PORT = process.env.PORT || 3000;
app.listen(PORT, () => console.log(`Server running on
http://localhost:${PORT}`));
```

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <title>News</title>
 </head>
 <body>
  <h1>News</h1>
  <ul>
   <% articles.forEach(x => { %>
   <li>
    <h2> <%= x.title %></h2>
    <p>Date: <%= x.date %></p>
    <p>
     <iframe
      width="560"
      height="315"
      src="<%= x.video %>
     ></iframe>
    </p>
    <p> <%= x.content %></p>
   </li>
   <% }) %>
  </ul>
 </body>
</html>
```

Example Partials

partial_views/header.ejs

```
<head>
  <link rel="stylesheet" href="/css/styles.css">
  <title>Portfolio</title>
</head>
```

partial_views/footer.ejs

```
<footer class="text-center text-muted py-3">
  <p>&copy; <%= new Date().getFullYear() %> Your Company Name</p>
  <p>All rights reserved.</p>
</footer>

<!-- Link to a specific JavaScript file for this footer or general scripts -->
<script src="/js/footer-scripts.js"></script>
```

```
<%- include("partial_views/header") %>
  <h1>Welcome to My Portfolio</h1>
  <p>Here you can find all my projects,
  education, and contact information.</p>
<%- include("partial_views/footer") %>
```

```
<%- include("partial_views/header") %>
  <h1>About Me</h1>
  <p>This is the about page,
  where I share my journey.</p>
<%- include("partial_views/footer") %>
```

```
<%- include("partial_views/header") %>
  <h1>Contact Me</h1>
  <p>Feel free to reach out
  via email or phone!</p>
<%- include("partial_views/footer") %>
```

```
app.get("/", (req, res) => {
  // This renders the indexpage.ejs file
  res.render("indexpage");
});

app.get("/about", (req, res) => {
  // Render aboutpage.ejs
  res.render("aboutpage");
});

app.get("/contact", (req, res) => {
  // Render contactpage.ejs
  res.render("contactpage");
});
```


When use EJS, JSX

- If your project is a simple web application that doesn't require a lot of dynamic interactions, and you want to keep things fast and simple on the server-side, EJS might be a good choice.
- If you're building a complex, dynamic web application with a lot of user interaction and want to take advantage of SPA, or you're ready to use React, JSX is the way to go.

Tags of EJS

<%	'Scriptlet' tag, for control-flow, no output
<%_	'Whitespace Slurping' Scriptlet tag, strips all whitespace before it
<%=	Outputs the value into the template (HTML escaped)
<%-	Outputs the unescaped value into the template
<%#	Comment tag, no execution, no output
<%%	Outputs a literal '<%'
%>	Plain ending tag
-%>	Trim-mode ('newline slurp') tag, trims following newline
_%>	'Whitespace Slurping' ending tag, removes all whitespace after it

Using EJS

- Install EJS with NPM:
  - npm install ejs
- Configure Express to Use EJS
  - Set up EJS as template engine

```
app.set('view engine', 'ejs');
```

  - Set up the directory containing EJS files

```
app.set('views', './templates');
```

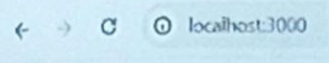
  - Using Template in Route Handler

```
app.get('/', (req, res) => {
    res.render('index', { name: 'MR. HUYNH NAM' });
});
```

- Create an EJS template file in the directory
  - Create a file named index.ejs with the content:

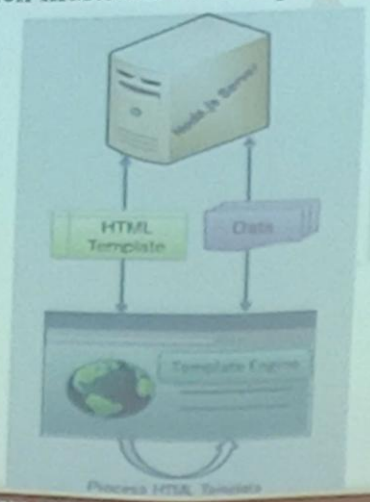
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-
width, initial-scale=1.0" />
    <title>Home Page</title>
  </head>
  <body>
    <h1>Hello, <%= name %>!</h1>
  </body>
</html>
```

- When accessing the path /, the browser displays "Hello, World!".



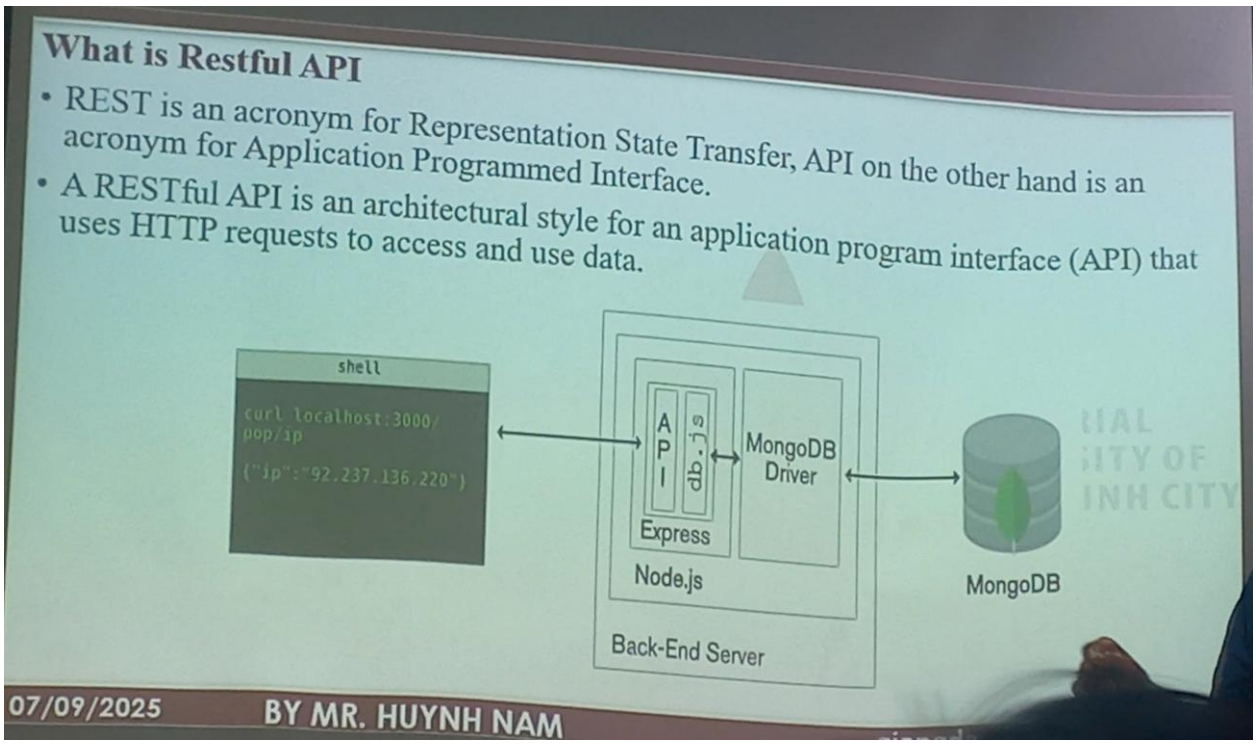
Templating with EJS

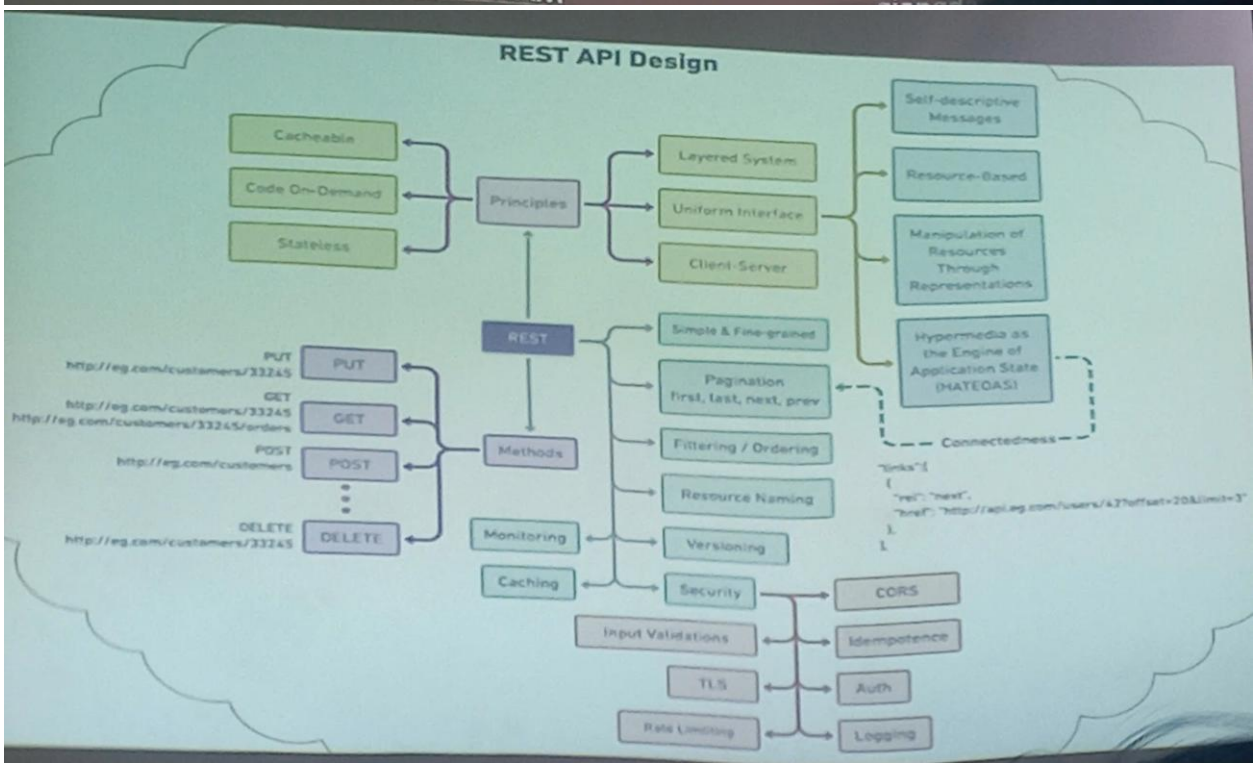
- A system that inserts data into your app's HTML template from the client side is known as a template or templating engine for Node.js.
- See the screenshot side, which illustrates the Template Engine Data Rendering Process.



Options of EJS

cache	Compiled functions are cached, requires filename
filename	Used by cache to key caches, and for includes
root	Set project root for includes with an absolute path (e.g, /file.ejs). Can be array to try to resolve include from multiple directories
views	An array of paths to use when resolving includes with relative paths.
context	Function execution context
compileDebug	When false no debug instrumentation is compiled
client	Returns standalone compiled function
delimiter	Character to use for inner delimiter, by default '%'
openDelimiter	Character to use for opening delimiter, by default '<'
closeDelimiter	Character to use for closing delimiter, by default '>'


What is Restful API
• REST is an acronym for Representation State Transfer, API on the other hand is an acronym for Application Programmed Interface.
• A RESTful API is an architectural style for an application program interface (API) that uses HTTP requests to access and use data.
shell
curl localhost:3000/pop/ip
{"ip":"92.237.136.220"}
API
db.js
MongoDB Driver
Express
Node.js
Back-End Server
MongoDB
07/09/2025
BY MR. HUYNH NAM


REST API Design
Cacheable
Code On-Demand
Stateless
Principles
Layered System
Uniform Interface
Client-Server
Self-descriptive Messages
Resource-Based
Manipulation of Resources Through Representations
Hypermedia as the Engine of Application State (HATEOAS)
REST
Simple & Fine-grained
Pagination first, last, next, prev
Filtering / Ordering
Resource Naming
Versioning
Security
CORS
Idempotence
Auth
Logging
Input Validations
TLS
Rate Limiting
Monitoring
Caching
Methods
PUT
GET
POST
DELETE
Connectedness

Implementing Population

- To actually merge these documents when querying, you use the populate() method provided by Mongoose.
- This method allows you to replace the specified paths in the document with document(s) from the path's referenced collection.
- Example of Population in a Query

```
Post.find().populate('author').exec((error, posts) => {
    if (error) throw error;
    console.log(posts);
// Each post will have a full 'author' document instead of
just an ObjectId.
});
```

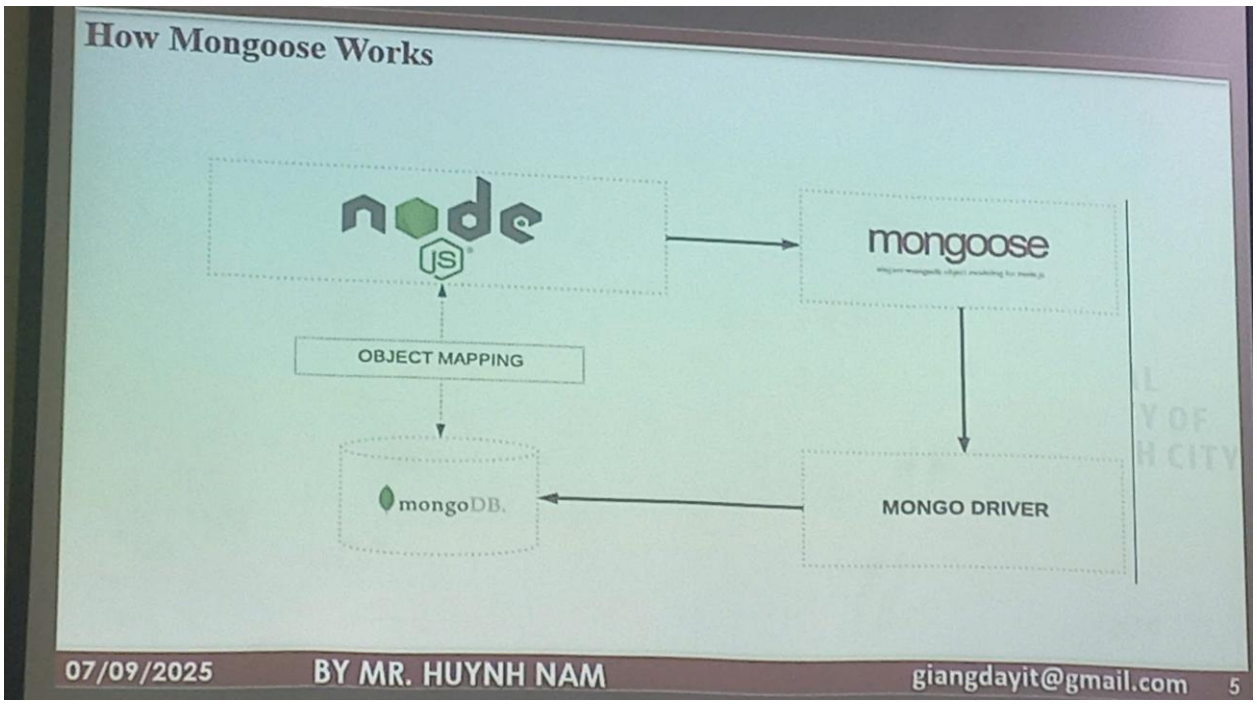
07/09/2025 BY MR. HUYNH NAM giangdayit@gmail.com

What is a schema

- A schema defines the structure of your collection documents. A Mongoose schema maps directly to a MongoDB collection.
- With schemas, we define each field and its data type. Permitted types are:
  - String
  - Number
  - Date
  - Buffer
  - Boolean
  - Mixed
  - ObjectId
  - Array
  - Decimal128
  - Map

```
const blog = new Schema({
    title: String,
    slug: String,
    published: Boolean,
    author: String,
    content: String,
    tags: [String],
    createdAt: Date,
    updatedAt: Date,
    comments: [{
        user: String,
        content: String,
        votes: Number
    }]
});
```

07/09/2025 BY MR. HUYNH NAM giangdayit@gmai


How Mongoose Works
node
JS
mongoose
OBJECT MAPPING
mongoDB.
MONGO DRIVER
07/09/2025
BY MR. HUYNH NAM
giangdayit@gmail.com
5


App
Database
Object
Student {
code
name
age
}
table
Document
Object
Mapping
RDBMS
(ORM)
Document
(ODM)
giangdayit@gmail.com
1

Connection strings

- The mongodb driver for Node.js is imported into the code with the require() function.
- The object of MongoClient class represents a database connection.
- You need to pass a connection string to its constructor.

```
const { MongoClient } = require('mongodb');
const client = new MongoClient(ConnectionString);
```

- Connection String format:
  - Standard Connection String Format
  - SRV Connection Format

Connecting to MongoDB

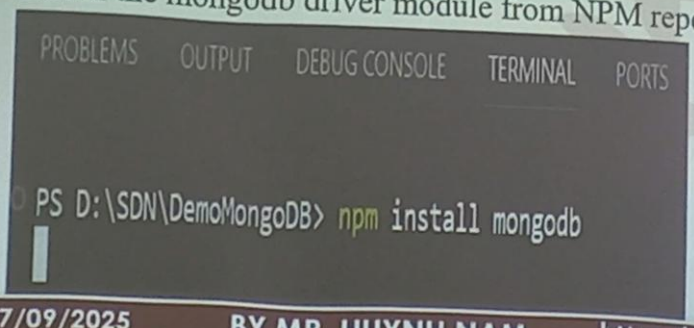
- We can establish connection with MongoDB server.
- Import the MongoClient class from the mongodb module with the require() statement.
- Call its connect() method by passing the MongoDB server URL.
- Example: Assuming that the above script is saved as app.js, run the application from command prompt.

```
PS D:\SDN\DemoMongoDB> node app.js
PS D:\SDN\DemoMongoDB> node app.js
Connected successfully to server
```

```
const { MongoClient } = require('mongodb');

// Connection URL
const url = 'mongodb://localhost:27017'; // If throw error then
replace localhost -> 127.0.0.1
const client = new MongoClient(url);

// Database Name
const dbName = 'myProject';

async function main() {
  // Use connect method to connect to the server
  await client.connect();
  console.log('Connected successfully to server');
  const db = client.db(dbName);
  const collection = db.collection('documents');

  // the following code examples can be pasted here...

  return 'done.';
}

main()
.then(console.log)
.catch(console.error)
.finally(() => client.close());
```

What Node MongoDB Driver

- Provides a high-level API for a Node application to interact with the MongoDB Server
- The Node MongoDB driver supports several operations that can be performed from a Node application:
  - Connecting to a MongoDB
  - Inserting, deleting, updating and querying documents
- Supports both callback based and promise based interactions
- Install the mongodb driver module from NPM repository with the following command

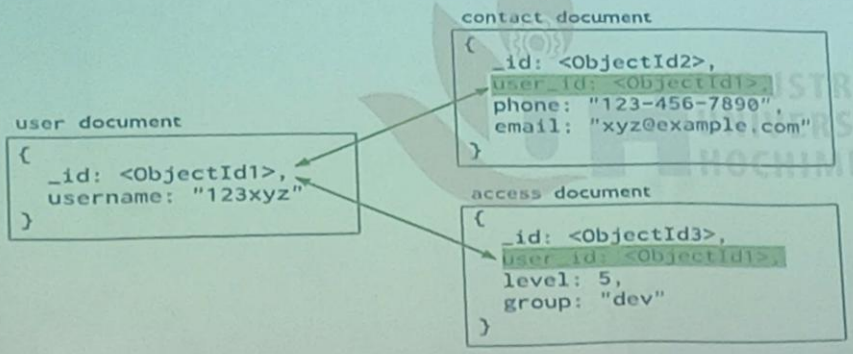


MongoDB tutorial

- English: https://www.geeksforgeeks.org/mongodb/mongodb-tutorial
- Việt Nam: https://janeto.gitbook.io/mongodb-toan-tap

Mapping Relationships

- A relationship is mapped according to the types of mapping that are made available in database relationships.
- This mapping will either be One-to-One or One-to-Many relationship mapping.
- MongoDB also made it possible to map relationships.

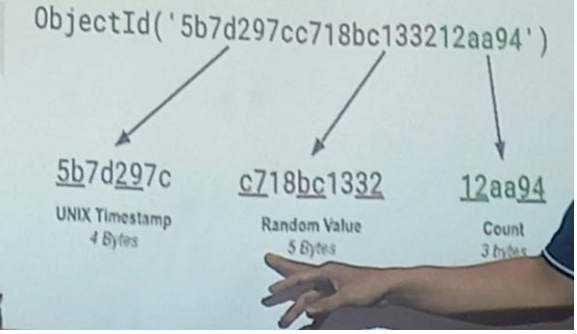


Basic commands in the MongoDB Shell

https://www.mongodb.com/developer/products/mongodb/cheat-sheet/

What is ObjectId in MongoDB

- Every document in the collection has an "_id" field that is used to uniquely identify the document in a particular collection it acts as the primary key for the documents in the collection. "_id" field can be used in any format and the default format is ObjectId of the document.
- It is a 12-byte Field Of BSON type
  - The first 4 bytes representing the Unix Timestamp of the document
  - The next 3 bytes are the machine Id on which the MongoDB server is running.
  - The next 2 bytes are of process id
  - The last Field is 3 bytes used for increment the objectid.



07/09/2025 BY MR. HUYNH NAM

MongoDB Atlas Cloud-hosted (Cloud-based)

Multi-region, multi-cloud
Run powerful and resilient apps that span multiple regions or clouds at once.

Serverless and elastic
Deploy a serverless database and only pay for the resources you use.

Native vector search capabilities
Build applications powered by semantic search and generative AI using native, full-featured vector database capabilities.

Always-on security
Secure data with built-in defaults for access and end-to-end encryption.

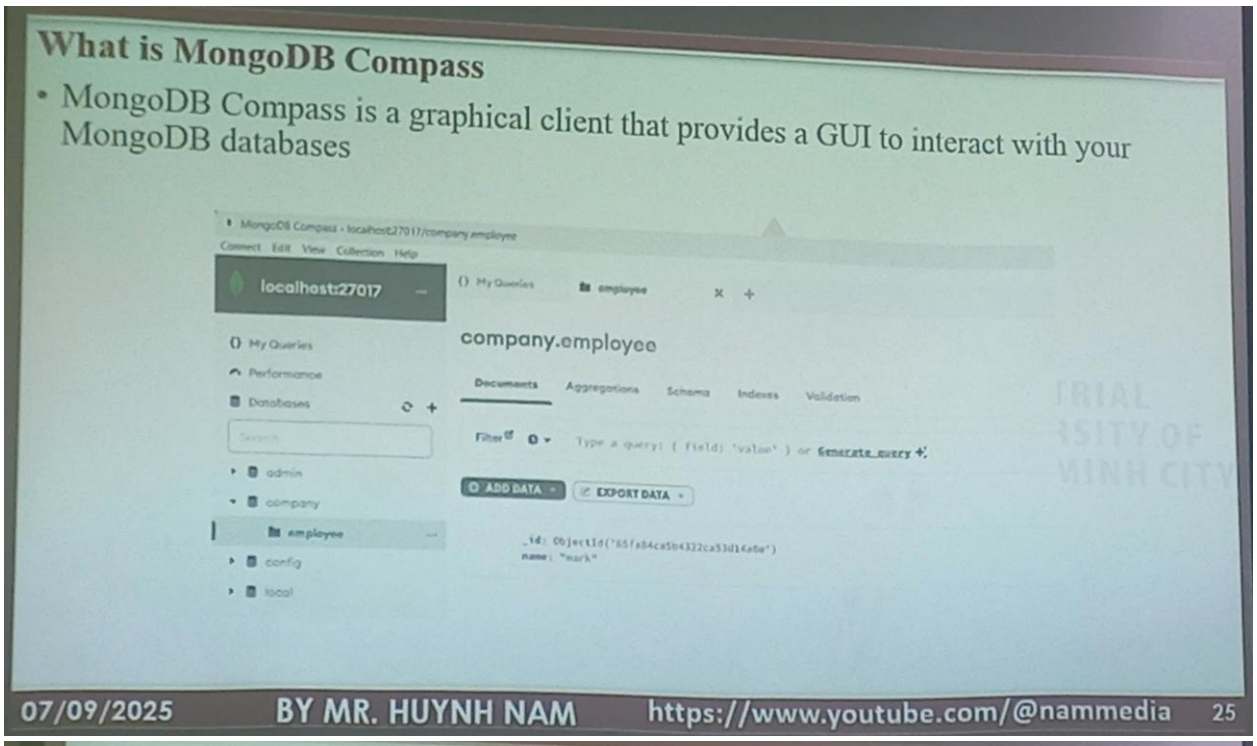
Performance advisor
Get on-demand index and schema design recommendations as your workload evolves.

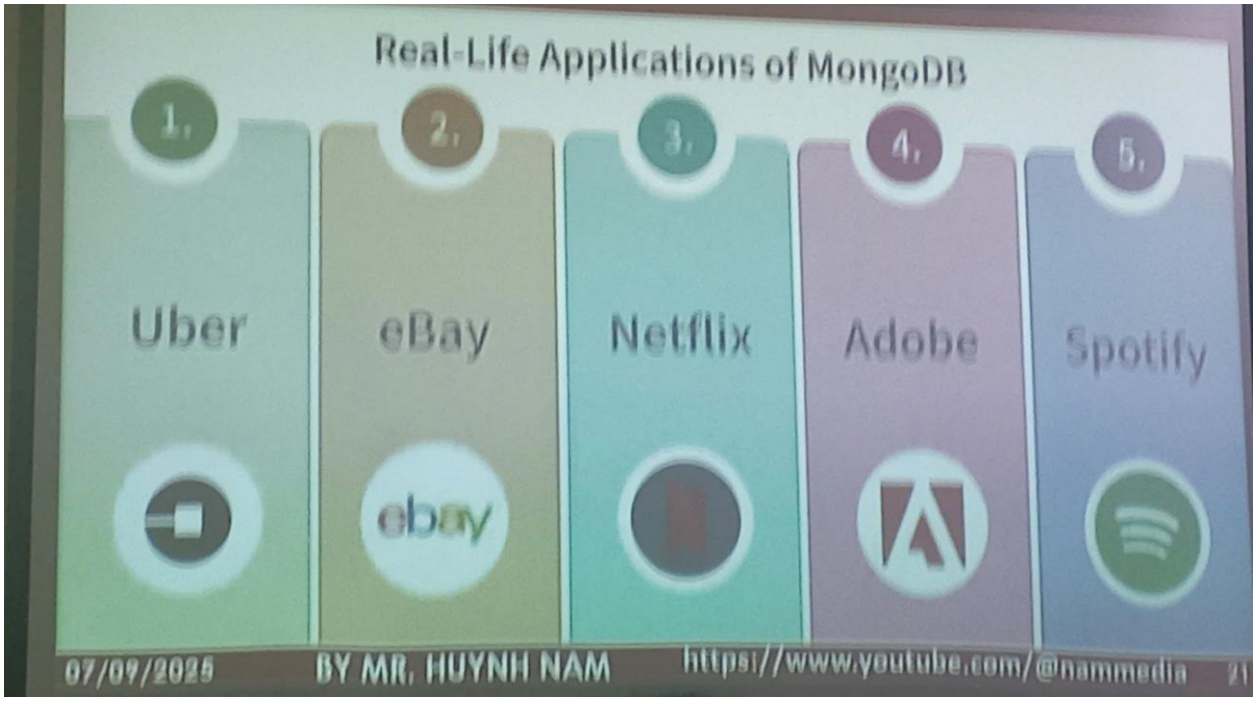
Automated data tiering
Set archival rules for cost-efficient data storage as your data estate grows.

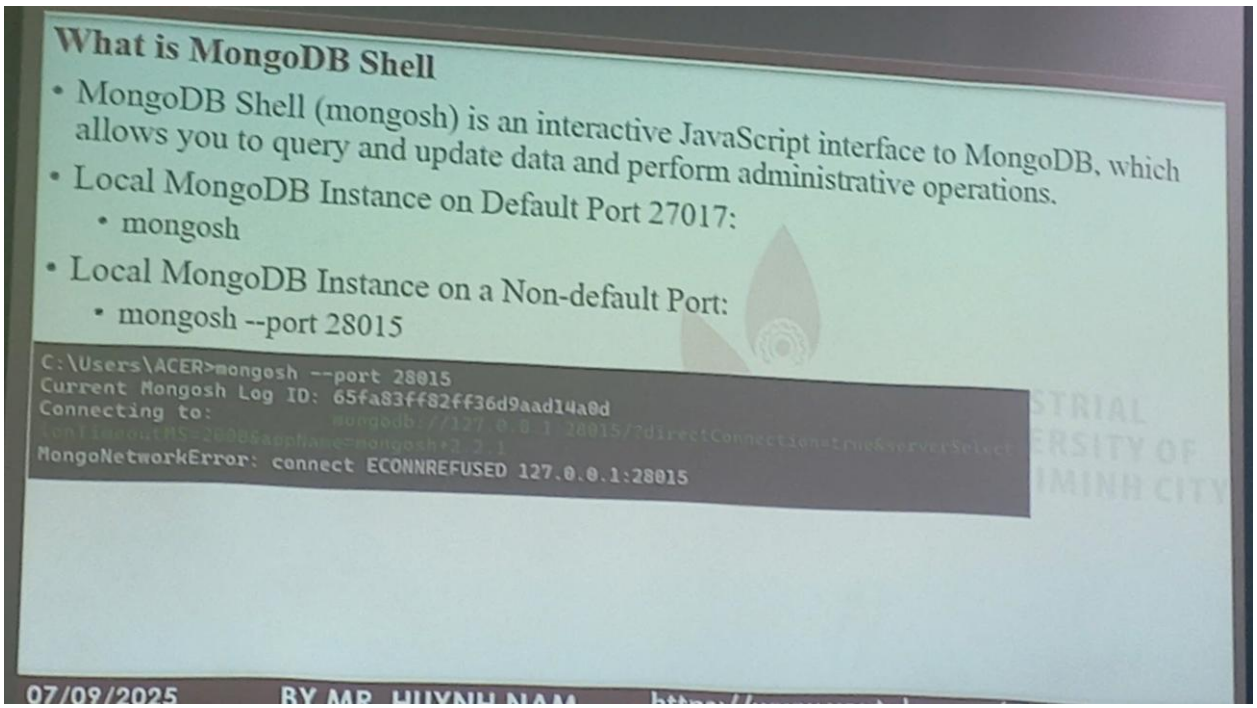
Continuous backups
Restore data to the precise moment you need with point-in-time recovery.

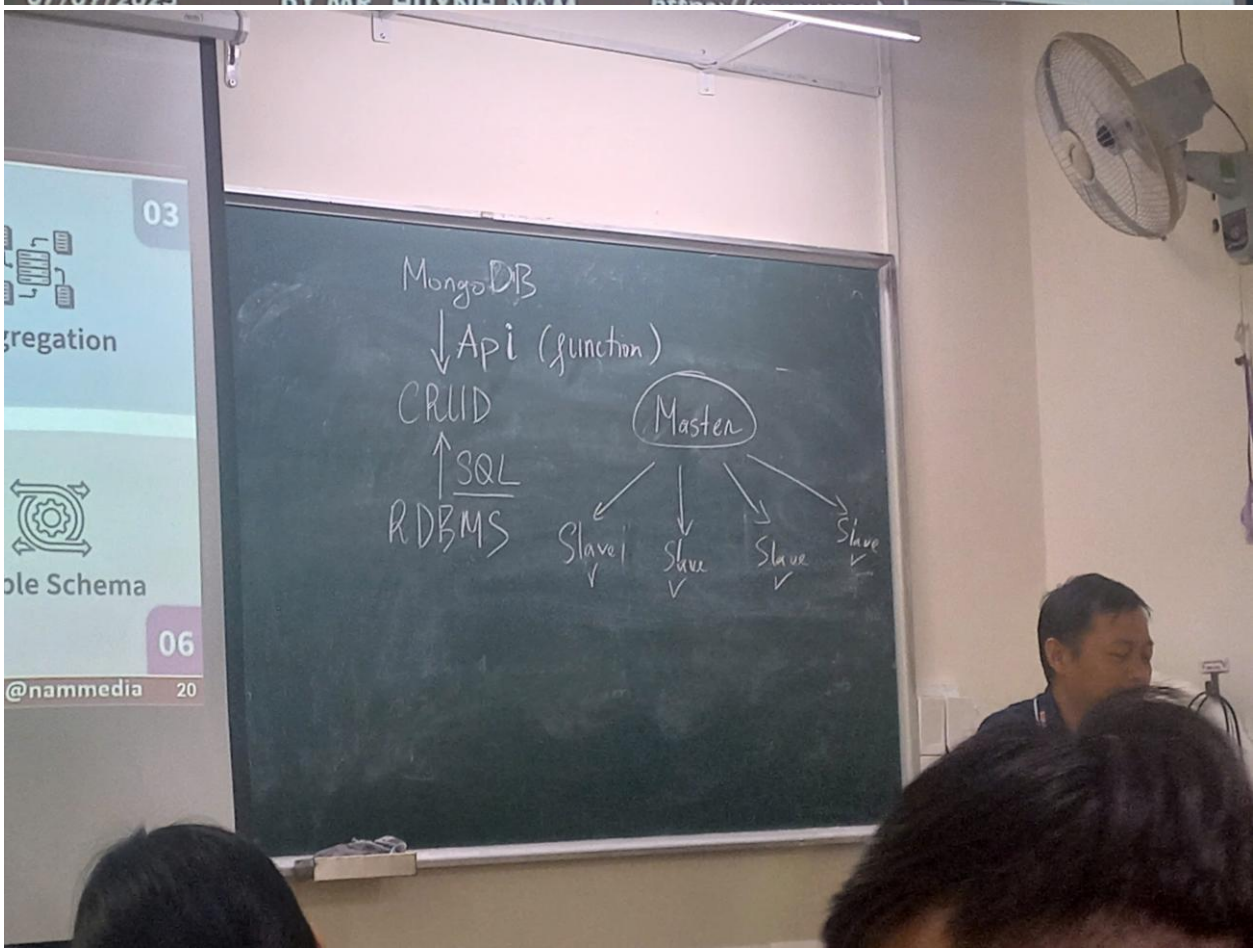
Workload isolation
Optimize local reads or run analytics with dedicated secondary nodes.

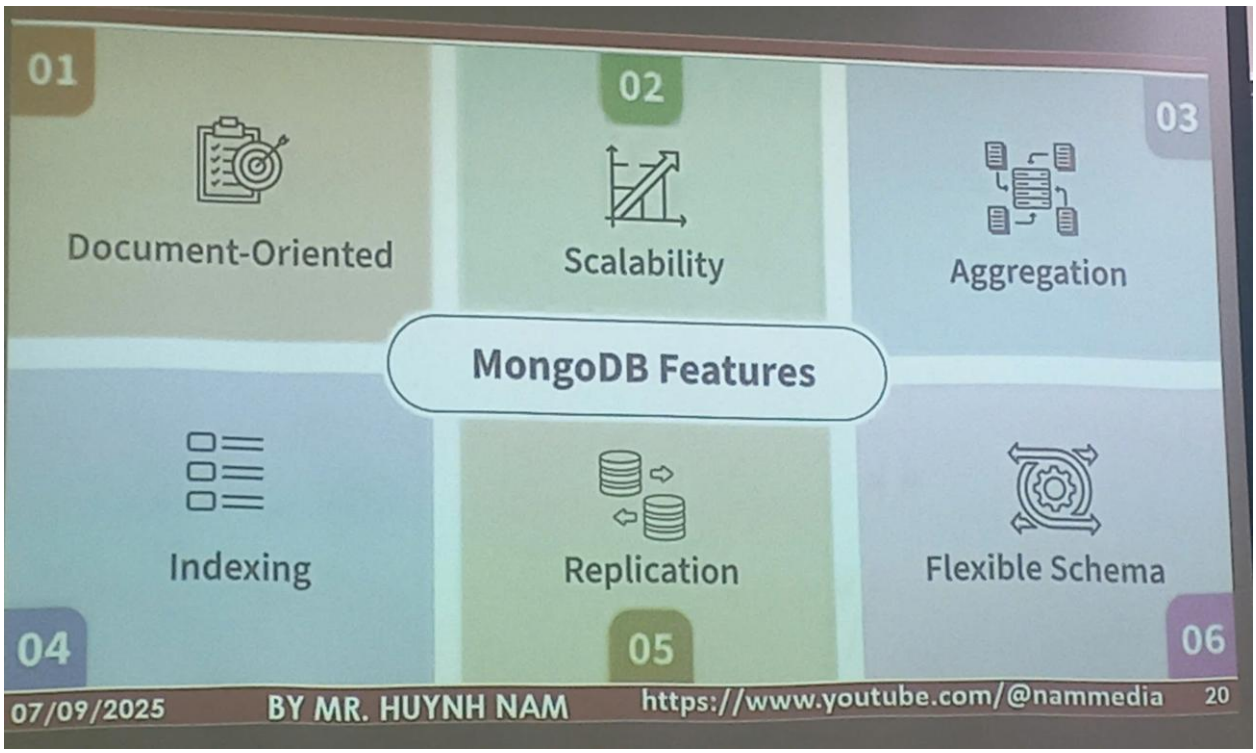
07/09/2025 BY MR. HUYNH NAM https://www.youtube.com/@nammedia 26

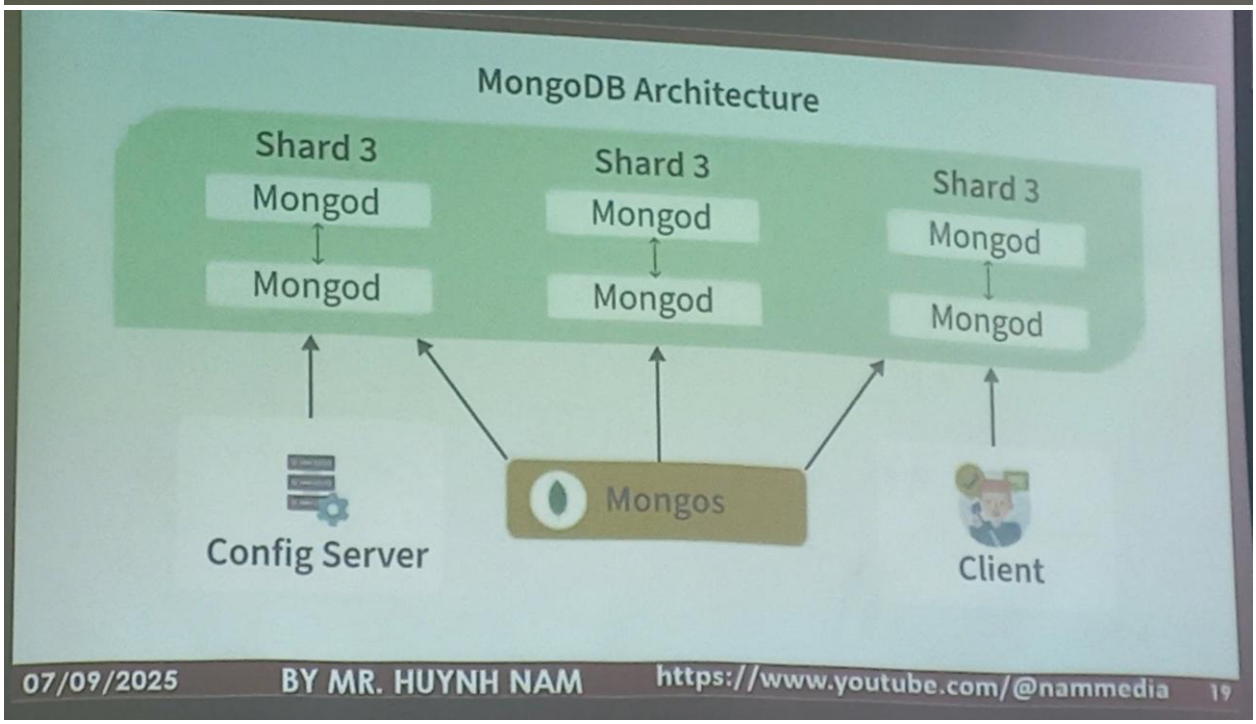

What is MongoDB Compass
• MongoDB Compass is a graphical client that provides a GUI to interact with your MongoDB databases
MongoDB Compass - localhost:27017/company.employee
Connect Edit View Collection Help
localhost:27017
My Queries
Performance
Databases
admin
company
employee
config
local
company.employee
Documents
Aggregations
Schema
Indexes
Validation
ADD DATA
EXPORT DATA
07/09/2025
BY MR. HUYNH NAM
https://www.youtube.com/@nammedia
25


Real-Life Applications of MongoDB
1.
Uber
2.
eBay
3.
Netflix
4.
Adobe
5.
Spotify
07/09/2025
BY MR. HUYNH NAM
https://www.youtube.com/@nammedia
21


What is MongoDB Shell
• MongoDB Shell (mongosh) is an interactive JavaScript interface to MongoDB, which allows you to query and update data and perform administrative operations.
• Local MongoDB Instance on Default Port 27017:
• mongosh
• Local MongoDB Instance on a Non-default Port:
• mongosh --port 28015
C:\Users\ACER>mongosh --port 28015
Current Mongosh Log ID: 65fa83ff82ff36d9aad14a0d
Connecting to:
MongoNetworkError: connect ECONNREFUSED 127.0.0.1:28015
07/09/2025

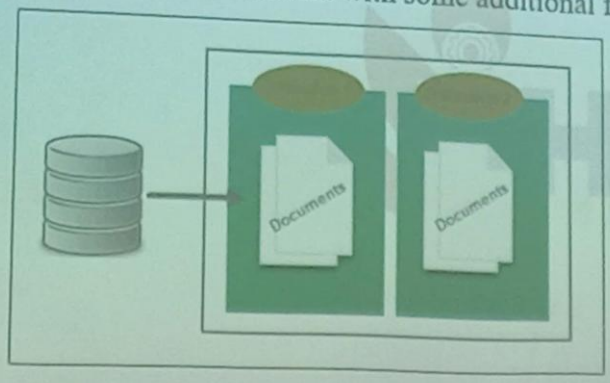

03
gregation
ble Schema
06
@nammedia 20
MongoDB
Api (function)
CRUD
SQL
RDBMS
Master
Slave
Slave
Slave
Slave


01
Document-Oriented
02
Scalability
03
Aggregation
MongoDB Features
Indexing
04
Replication
05
Flexible Schema
06
07/09/2025
BY MR. HUYNH NAM
https://www.youtube.com/@nammedia
20


MongoDB Architecture
Shard 3
Mongod
Mongod
Shard 3
Mongod
Mongod
Shard 3
Mongod
Mongod
Config Server
Mongos
Client
07/09/2025
BY MR. HUYNH NAM
https://www.youtube.com/@nammedia
19

MongoDB

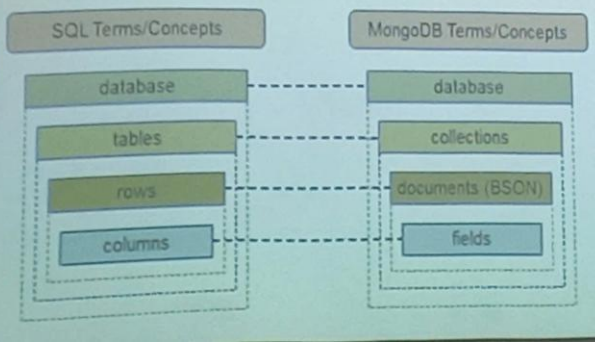
- Document Database
  - Server can support multiple databases
  - A database consists of a set of collections
  - A collection is a set of documents
  - Document is effectively a JSON document with some additional features



Leveraging your knowledge of traditional databases

- collection = like database table
- document = entry /row in collection = like database row
- field = like a column but, --there is no ordering or notion of required
- field _id = primary key

RDBMS	as	MongoDB
Database	as	Database
Table, View	as	Collection
Row	as	Document (JSON, BSON)
Column	as	Field
Index	as	Index
Join	as	Embedded Document
Foreign Key	as	Reference
Partition	as	Shard



Difference between SQL and NoSQL databases

SQL vs. NoSQL: Comparison

SQL		NoSQL
Relational	Model	Non-relational
Structured tables	Data	Semi-structured
Strict schema	Flexibility	Dynamic schema
ACID	Transactions	Mostly BASE, few ACID
Strong	Consistency	Eventual to Strong
Consistency prioritized	Availability	Basic Availability
Vertically by upgrading hardware	Scale	Horizontally by data partitioning

ml4devs.com/datastores

- **Key-Value** → Dictionary or Hash Table
- **Wide Column** → 2-D Versioned Key-Value
- **Document** → Nested Objects (XML, JSON, YAML)
- **Graph** → Entity-Relationships

scgupta.me
twitter.com/scgupta
linkedin.com/in/scgupta

NoSQL Databases

Column-Based Store

- Data is stored in columns, which are grouped into families, and these families are further grouped into more columns.

Row-oriented

ID	Name	Grade	GPA
001	John	Senior	4.00
002	Karen	Freshman	3.67
003	Bill	Junior	3.33

Column-oriented

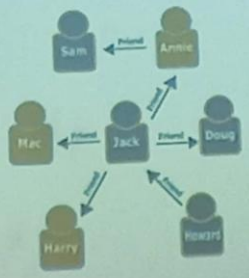
Name	ID
John	001
Karen	002
Bill	003

Grade	ID
Senior	001
Freshman	002
Junior	003

GPA	ID
4.00	001
3.67	002
3.33	003

Graph-Based Store

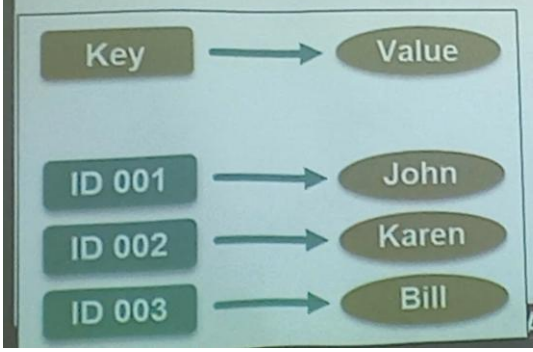
- Graph or network data models essentially treat the relationship between any two pieces of information as being just as important as the information itself.



NoSQL Databases

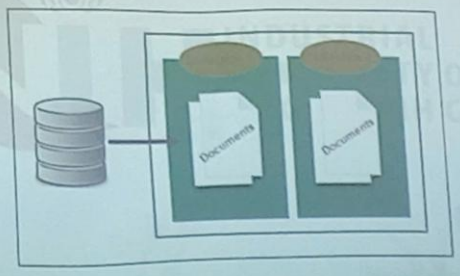
Key-Value Store

- Key-value stores use key values with pointers to store data
- This pointer is unique and links directly to a piece of information, which can be anything you'd like it to be.

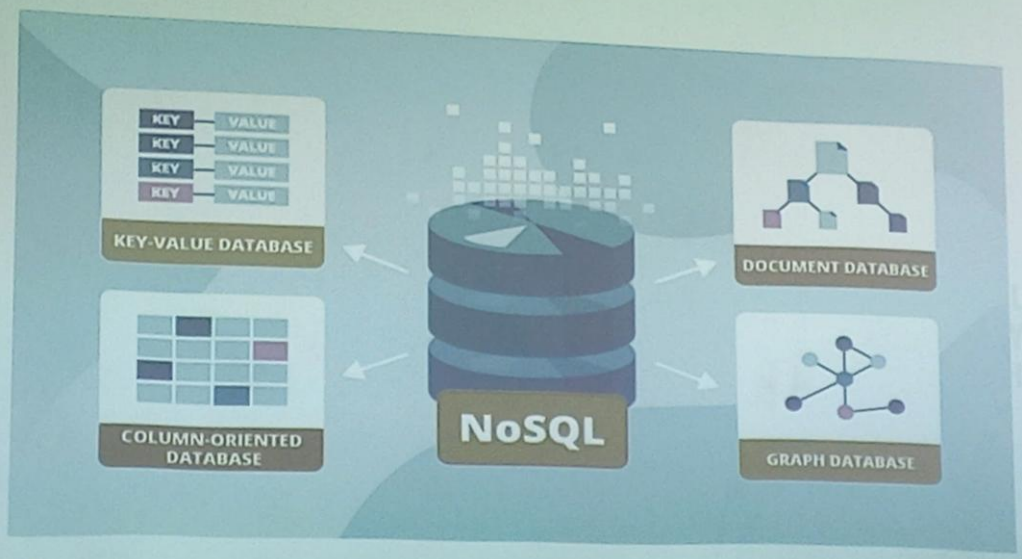


Document-Based Store

- All data is stored in one table, so there's no need for cross-referencing and instead of storing information in a table, it's stored in a document.

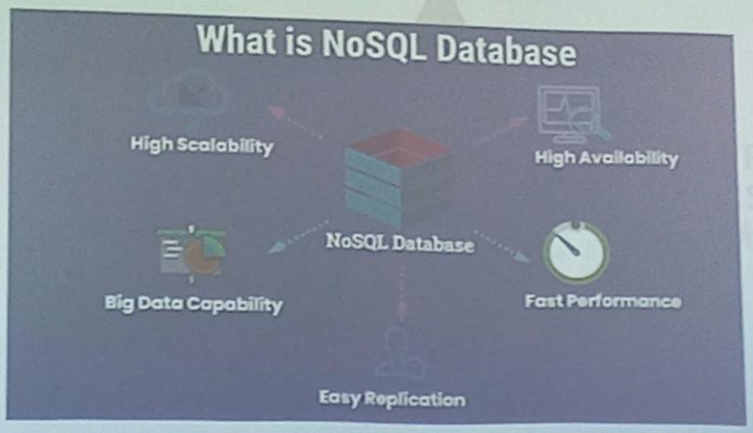


Four Types of NoSQL Databases



What is NoSQL Database

- No use a relational data modeling technique
- High efficiency and speed in terms of creating up to millions of queries a second
- Horizontally scalable



Why is Data Serialization Important for Distributed Systems

- In some distributed systems, data and its replicas are stored in different partitions on multiple cluster members. If data is not present on the local member, the system will retrieve that data from another member. This requires serialization for use cases such as:
  - Adding key/value objects to a map
  - Putting items into a queue, set, or list
  - Sending a lambda function to another server
  - Processing an entry within a map
  - Locking an object
  - Sending a message to a topic

Create HTTP Server in NodeJS

- Install http

```
PS D:\SDN\Demo> npm i http

added 1 package, and audited 2 packages in 1s
```

- Check dependency in package.json

package.json

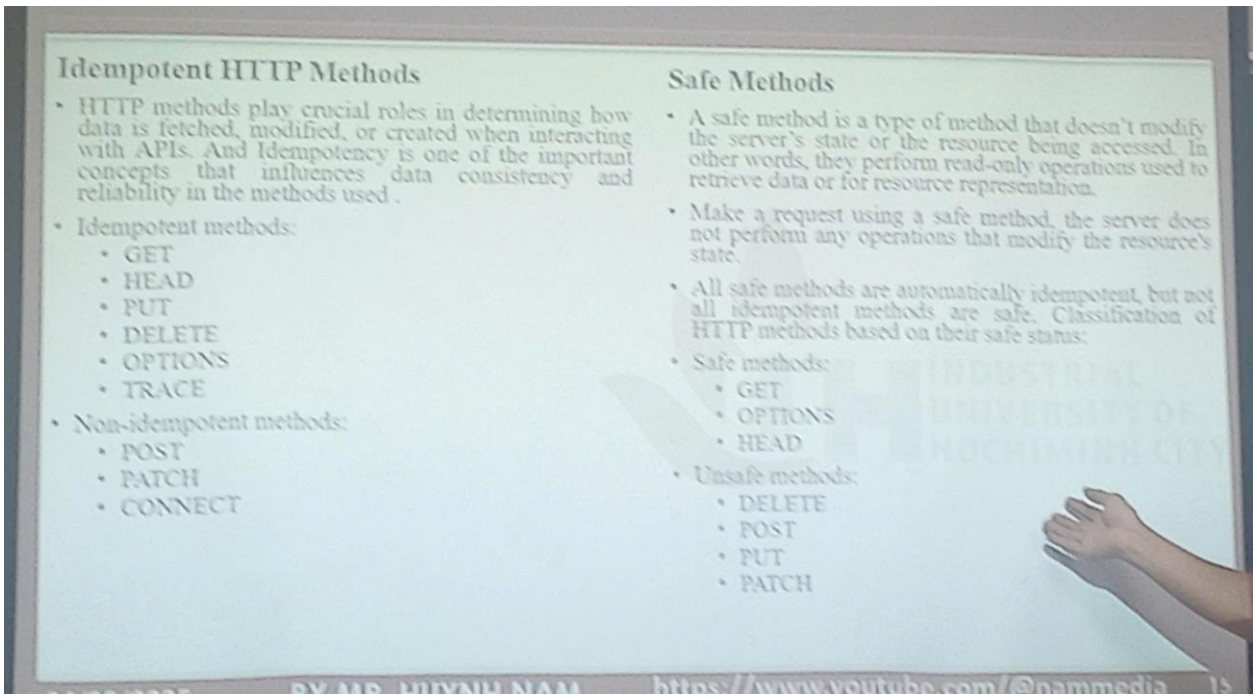
```
{
  "name": "demo",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "http": "^0.0.1-security"
  }
}
```

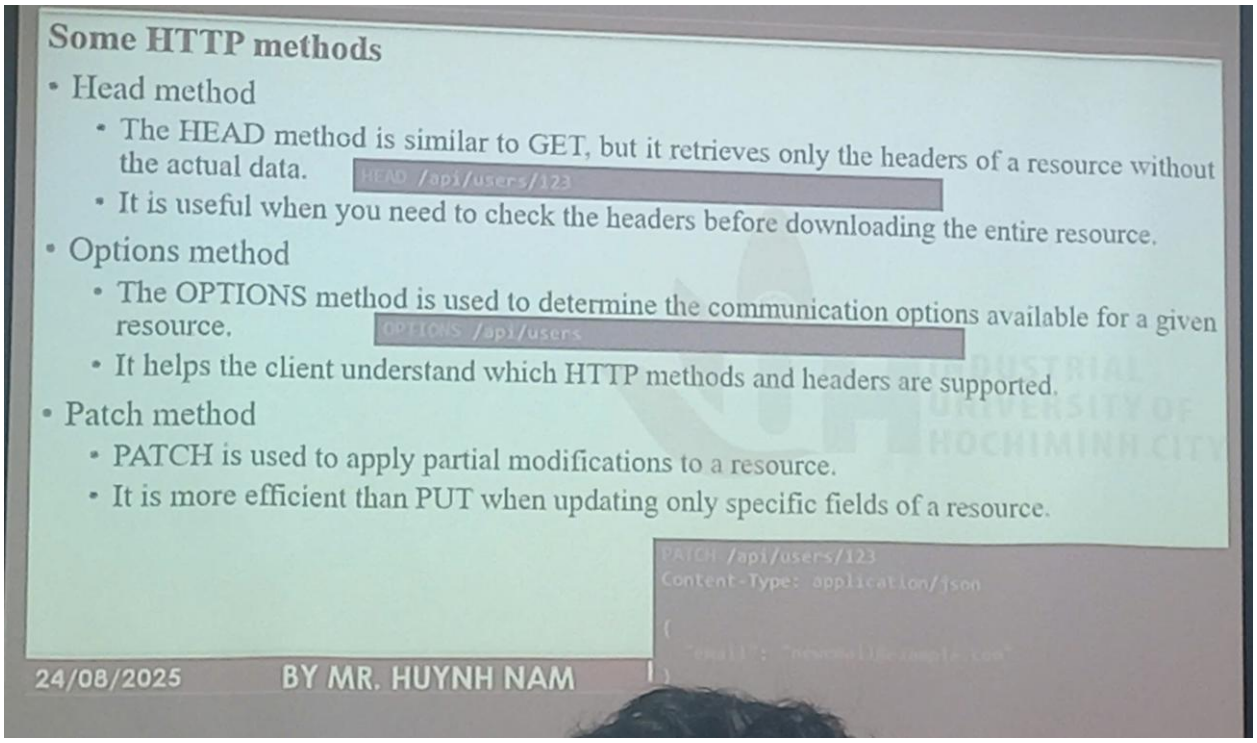
- Building HTTP-based applications in Node.js, the built-in http/https modules are the ones you will interact with.
- The http module is very low-level – creating a complex web application using the snippet above is very time-consuming. This is the reason why we usually pick a framework to work with for big projects.

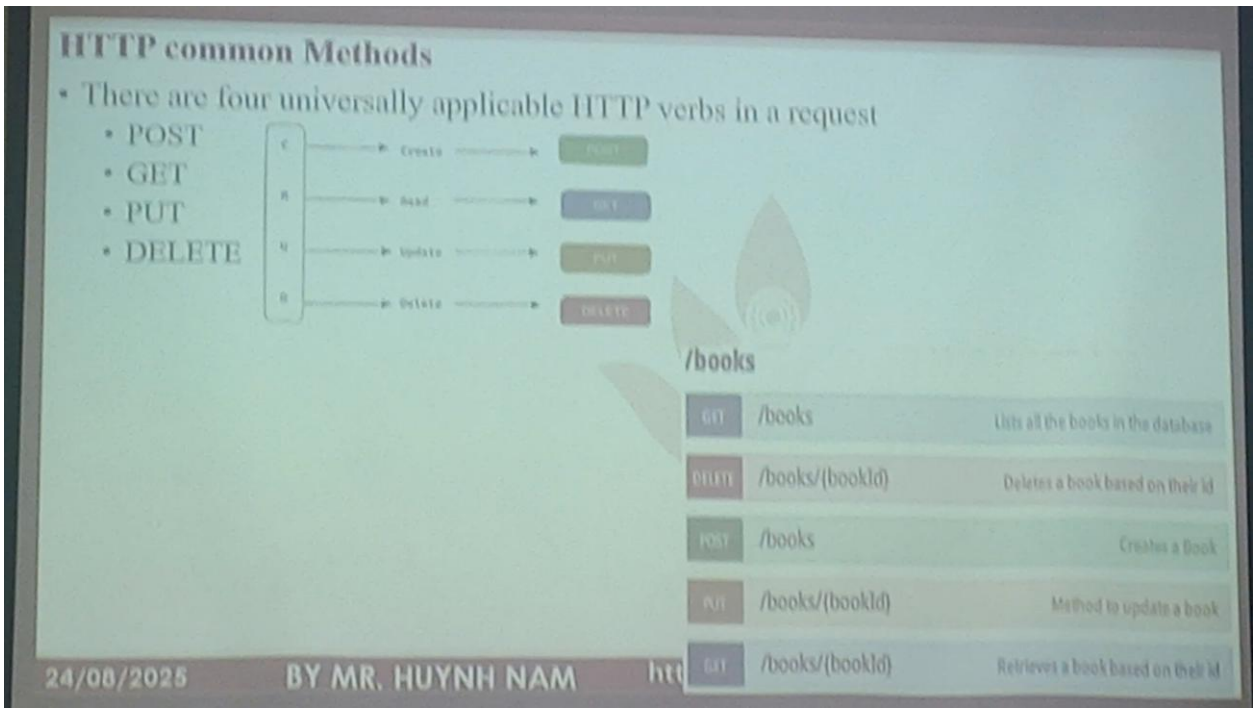
```
//index.js
var http = require("http");
const hostname = "localhost";
const port = 3000;
const server = http.createServer(function (request, response) {
    //Send the http header
    console.log(request.headers);
    //HTTP Status: 200 : OK
    response.statusCode = 200;
    //Content Type: text/plain
    response.writeHead(200, { 'Content-Type': 'text/plain' });
    //Send the response body as "Hello World"
    response.end("Hello World!\n");
}).listen(port);
//Console will print the message
console.log("Server running at http://${hostname}:${port}/");
```

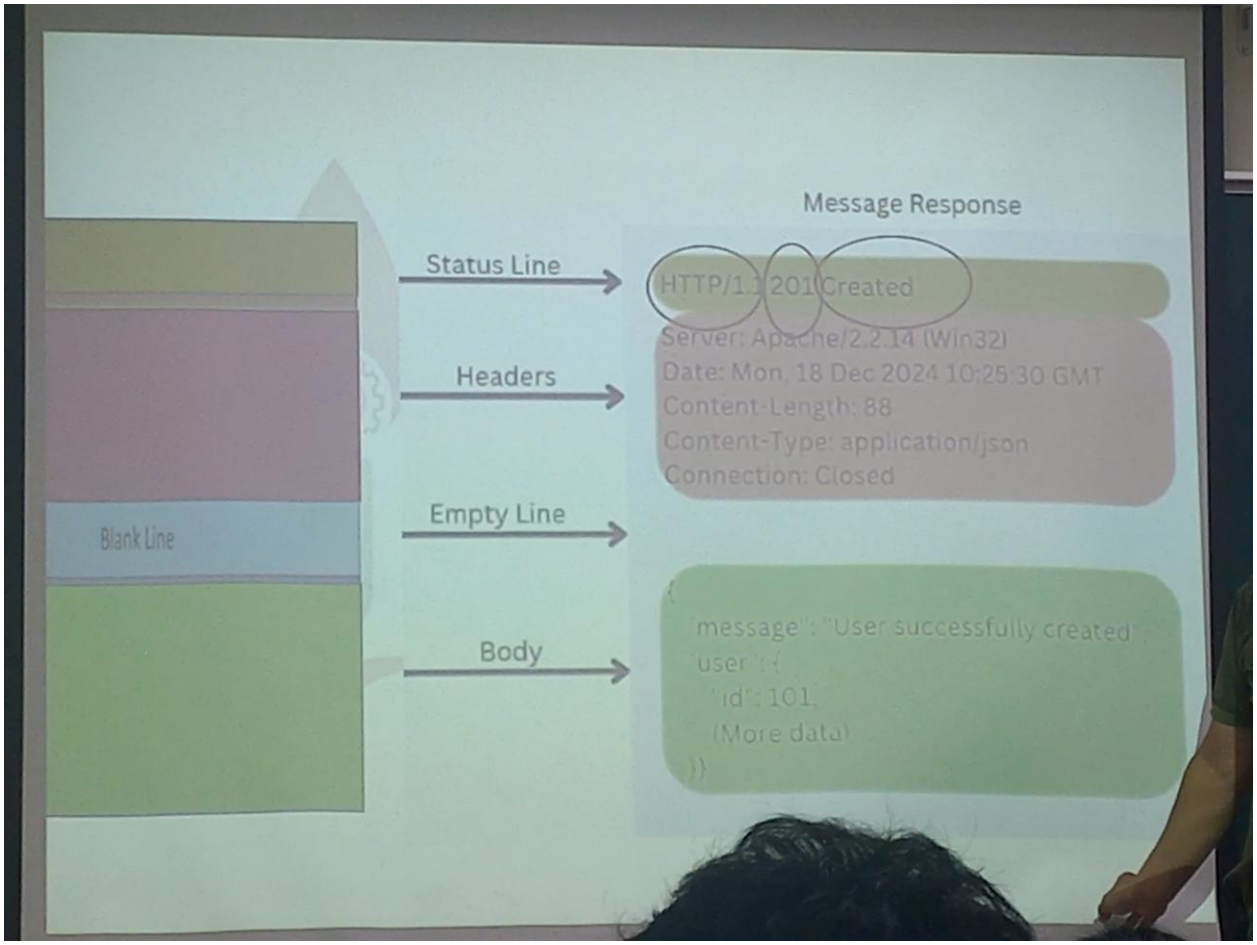
Status Codes

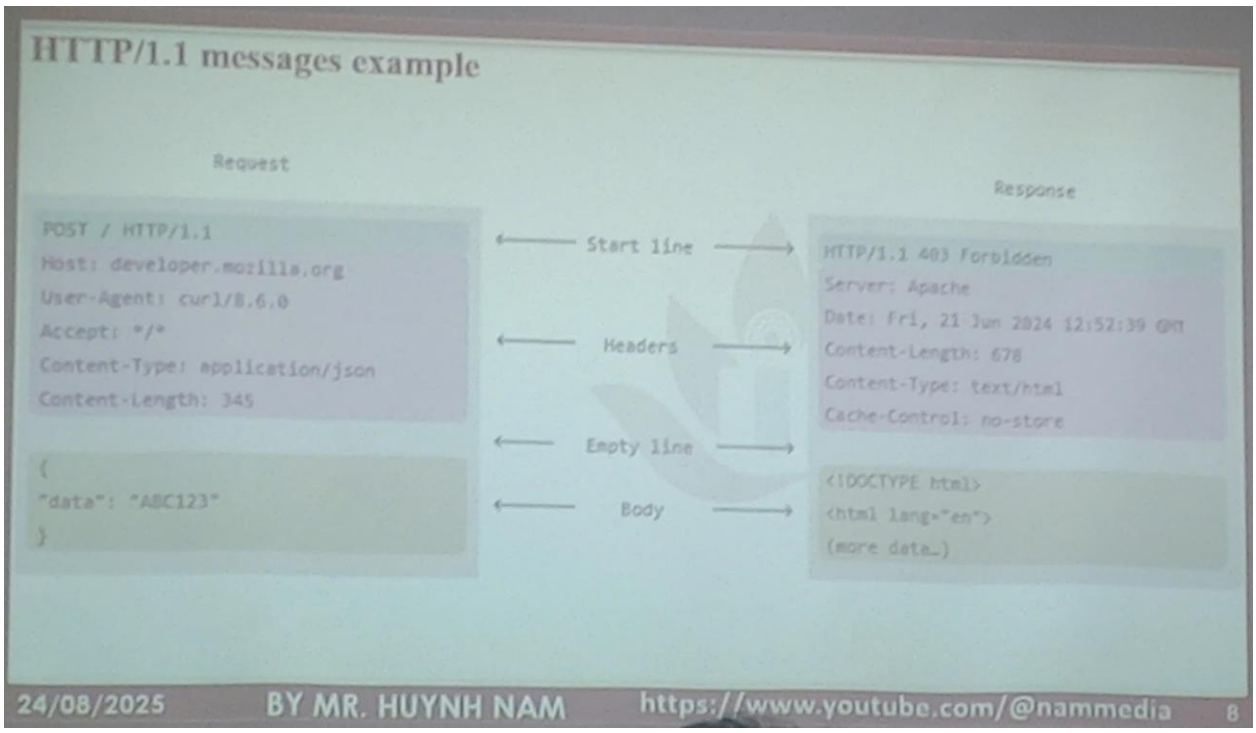
Code	Meaning
200	OK
201	Created
301	Moved Permanently
304	Not Modified
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
422	Unprocessable Entry
500	Internal Server Error
505	HTTP Version Not Supported

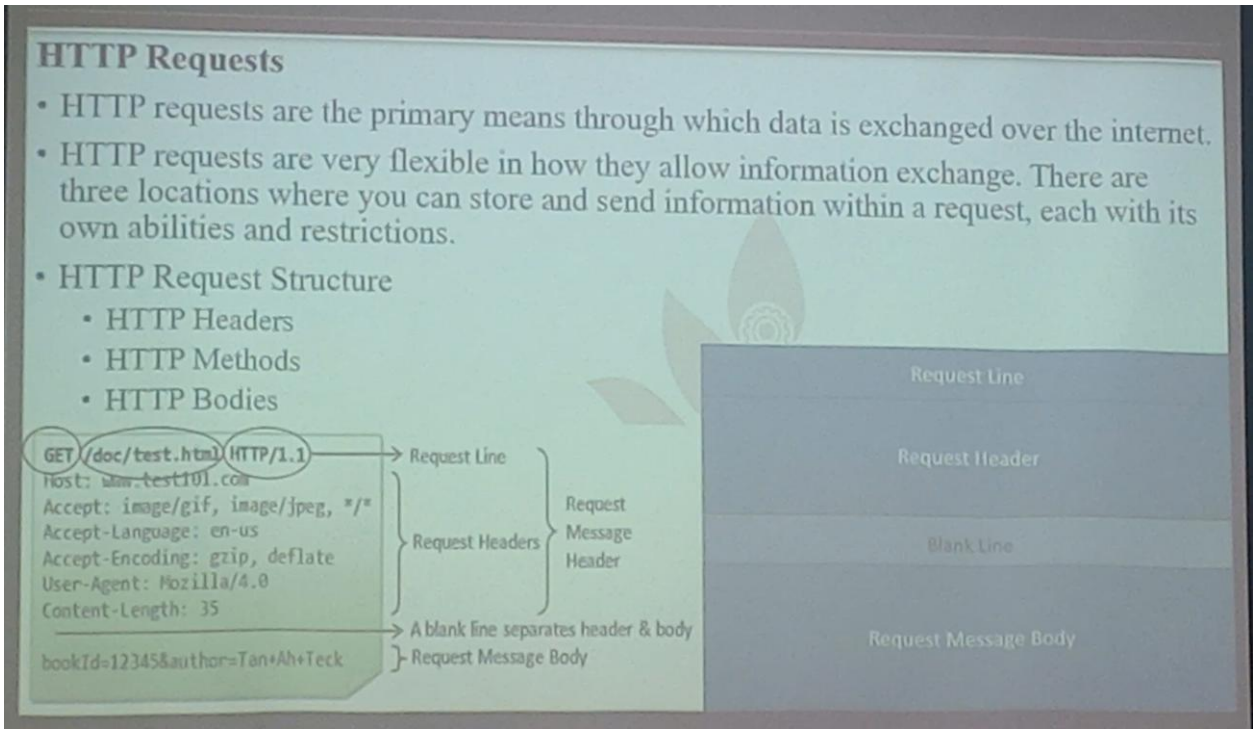

Idempotent HTTP Methods
HTTP methods play crucial roles in determining how data is fetched, modified, or created when interacting with APIs. And Idempotency is one of the important concepts that influences data consistency and reliability in the methods used .
Idempotent methods:
GET
HEAD
PUT
DELETE
OPTIONS
TRACE
Non-idempotent methods:
POST
PATCH
CONNECT
Safe Methods
A safe method is a type of method that doesn't modify the server's state or the resource being accessed. In other words, they perform read-only operations used to retrieve data or for resource representation.
Make a request using a safe method, the server does not perform any operations that modify the resource's state.
All safe methods are automatically idempotent, but not all idempotent methods are safe. Classification of HTTP methods based on their safe status:
Safe methods:
GET
OPTIONS
HEAD
Unsafe methods:
DELETE
POST
PUT
PATCH


Some HTTP methods
Head method
The HEAD method is similar to GET, but it retrieves only the headers of a resource without the actual data.
HEAD /api/users/123
It is useful when you need to check the headers before downloading the entire resource.
Options method
The OPTIONS method is used to determine the communication options available for a given resource.
OPTIONS /api/users
It helps the client understand which HTTP methods and headers are supported.
Patch method
PATCH is used to apply partial modifications to a resource.
It is more efficient than PUT when updating only specific fields of a resource.
PATCH /api/users/123
Content-Type: application/json
{
"email": "newemail@example.com"
}
24/08/2025
BY MR. HUYNH NAM


HTTP common Methods
There are four universally applicable HTTP verbs in a request
POST
GET
PUT
DELETE
/books
GET /books Lists all the books in the database
DELETE /books/{bookId} Deletes a book based on their id
POST /books Creates a Book
PUT /books/{bookId} Method to update a book
GET /books/{bookId} Retrieves a book based on their id
24/08/2025
BY MR. HUYNH NAM

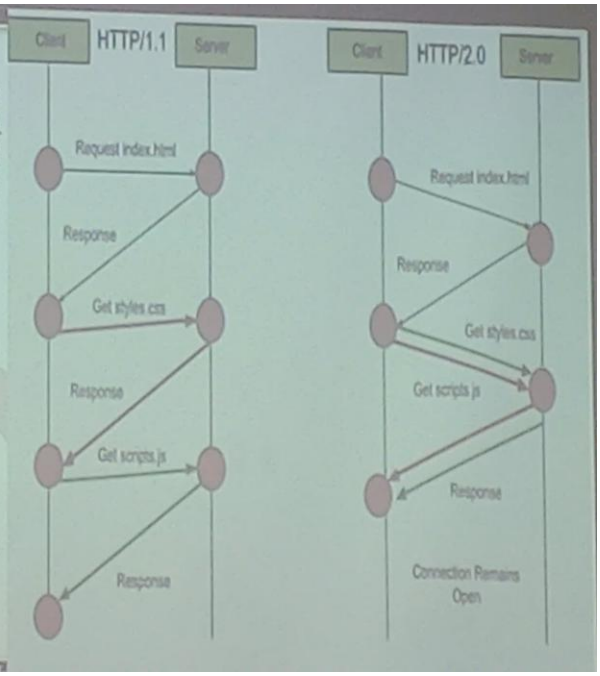

Message Response
Status Line
HTTP/1.1 201 Created
Headers
Server: Apache/2.2.14 (Win32)
Date: Mon, 18 Dec 2024 10:25:30 GMT
Content-Length: 88
Content-Type: application/json
Connection: Closed
Empty Line
Blank Line
Body
{
"message": "User successfully created",
"user": {
"id": 101,
(More data)
}}


HTTP/1.1 messages example
Request
Response
POST / HTTP/1.1
Host: developer.mozilla.org
User-Agent: curl/8.6.0
Accept: */*
Content-Type: application/json
Content-Length: 345
{
"data": "ABC123"
}
Start line
Headers
Empty line
Body
HTTP/1.1 403 Forbidden
Server: Apache
Date: Fri, 21 Jun 2024 12:52:39 GMT
Content-Length: 678
Content-Type: text/html
Cache-Control: no-store
<!DOCTYPE html>
<html lang="en">
(more data…)
24/08/2025
BY MR. HUYNH NAM
https://www.youtube.com/@nammedia
8


HTTP Requests
• HTTP requests are the primary means through which data is exchanged over the internet.
• HTTP requests are very flexible in how they allow information exchange. There are three locations where you can store and send information within a request, each with its own abilities and restrictions.
• HTTP Request Structure
• HTTP Headers
• HTTP Methods
• HTTP Bodies
GET /doc/test.html HTTP/1.1
Host: www.test101.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Content-Length: 35
bookId=12345&author=Tan+Ah+Teck
Request Line
Request Headers
Request Message Header
A blank line separates header & body
Request Message Body
Request Line
Request Header
Blank Line
Request Message Body

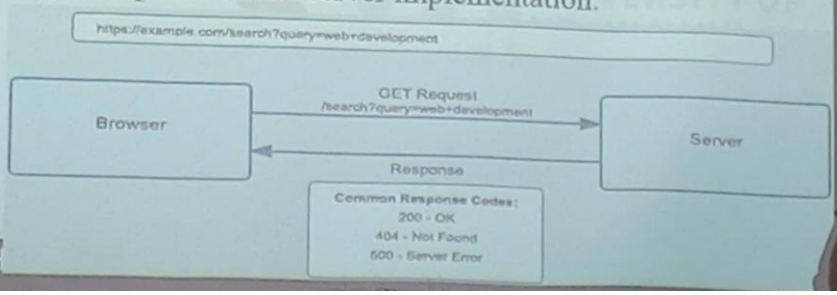
HTTP (Hypertext Transfer Protocol)

- HTTP messages are the mechanism used to exchange data between a server and a client in the HTTP protocol. There are two types of messages: requests sent by the client to trigger an action on the server, and responses, the answer that the server sends in response to a request.
- HTTP/1.1
  - Modification was made in 2014
  - Each TCP/IP connection is limited to one open request in this version
  - HTTP/0.9 was the protocol's first version. Currently, this version is entirely obsolete.
- HTTP/2.0
  - The most recent iterations of Chrome, Firefox, Safari, and Edge all support HTTP/2.0.
  - When using this HTTP version, the client can send numerous queries at once.
  - It accelerates the speed at which a page loads.



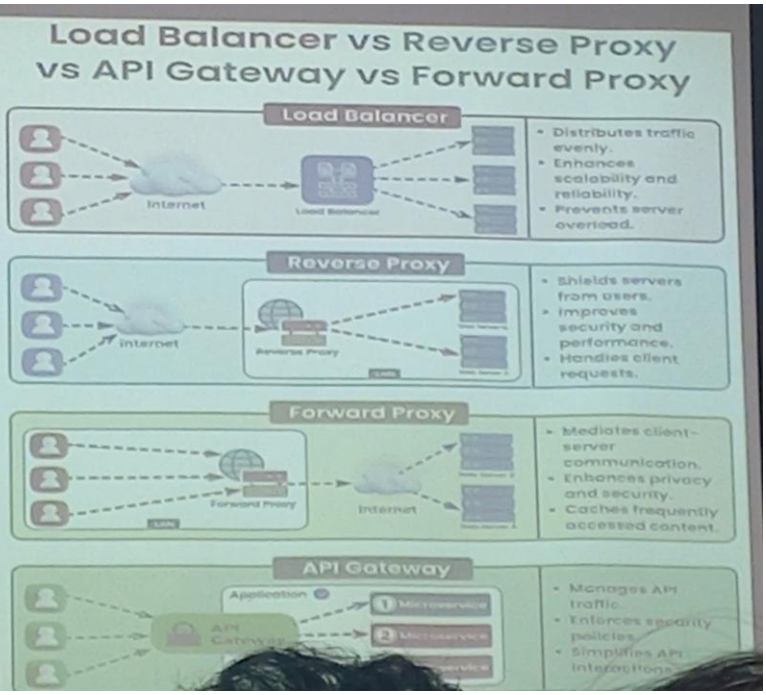
What is the HTTP (Hypertext Transfer Protocol)

- It is the foundation of communication between clients and servers on the web
- Stateless, Application-layer protocol for dealing with distributed services.
- Requests are sent across the TCP/IP layer, and responses are returned.
- HTTP Versions.
- HTTP – defines several methods (referred to as "verbs") that indicates the desired action to be performed on a resource. The resource is specified by the **URI** (Uniform Resource Identifier), more commonly, the **URL**. This resource may be pre-existing data or data that is generated dynamically, it depends on the server implementation.



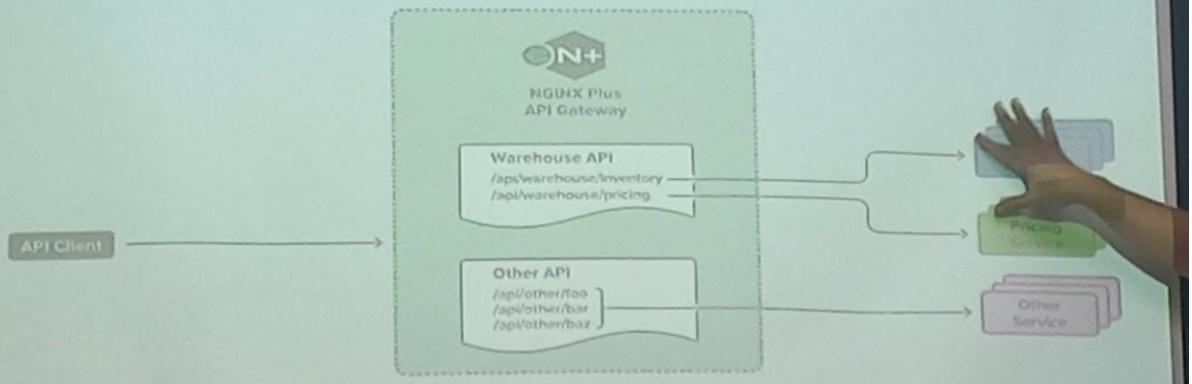
24/08/2025 BY MR. H

- Load Balancer vs Reverse Proxy vs Forward Proxy vs API Gateway all handle incoming requests before they reach backend services, but they differ in their primary function and the layer at which they operate.

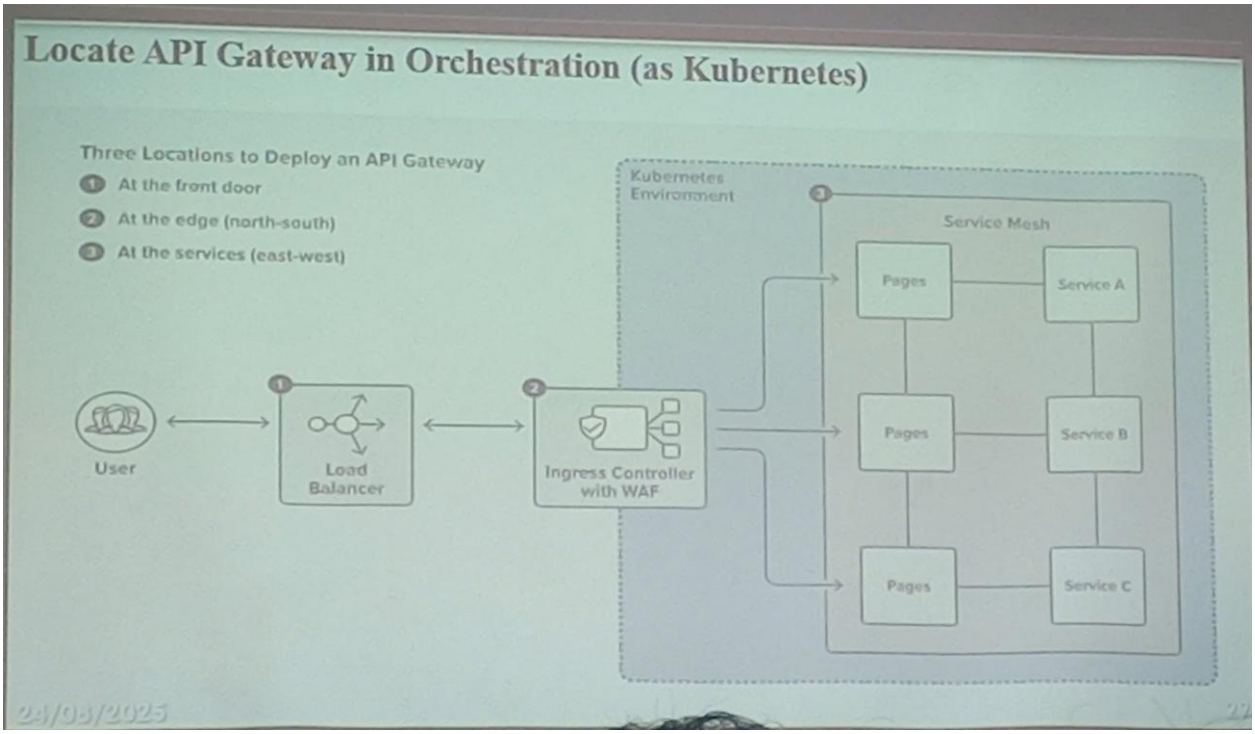


API gateway

- An API gateway is a management tool that acts as a single entry-point for all API calls from clients to backend services, particularly in a microservices architecture. It sits between the client applications (e.g., mobile apps, web browsers) and the various backend services (e.g., microservices, traditional monolithic applications).



Locate API Gateway in Orchestration (as Kubernetes)



24/08/2025

REST APIs versus RESTful API

- **REST API** follows many of the principles of REST (such as using standard HTTP methods), it may not necessarily implement all the constraints as Stateless, Flexible Data Formats, Basic HTTP Methods and Simpler Design.
- Use case: Basic Web Applications, Mobile Applications
- Examples of REST APIs include:
  - Amazon Product Advertising REST API.
  - Flickr REST API.
  - YouTube REST API.

- A **RESTful API** has all the features of the REST architecture, but it *differs slightly in that it has a transport protocol* and *keeps cached data available at all times rather than deleting it when the API is not in use*. Due to its unique features, *a RESTful API tends to be more flexible yet less secure.*
- Features of RESTful API
  - REST Principles: RESTful APIs strictly follow the six architectural constraints of REST (Stateless, Client-Server, Uniform Interface, Cacheable, Layered System, and Code on Demand).
  - Use of HTTP Methods: RESTful APIs consistently use the appropriate HTTP methods to interact with resources (e.g., GET for retrieval, POST for creation, PUT for updates, DELETE for removal).
  - Resource Identification via URIs: Each resource is accessible via a unique URI (Uniform Resource Identifier), making it easier to access and manipulate data.
  - Separation of Concerns: RESTful APIs promote a clear separation between client and server, allowing them to evolve independently.

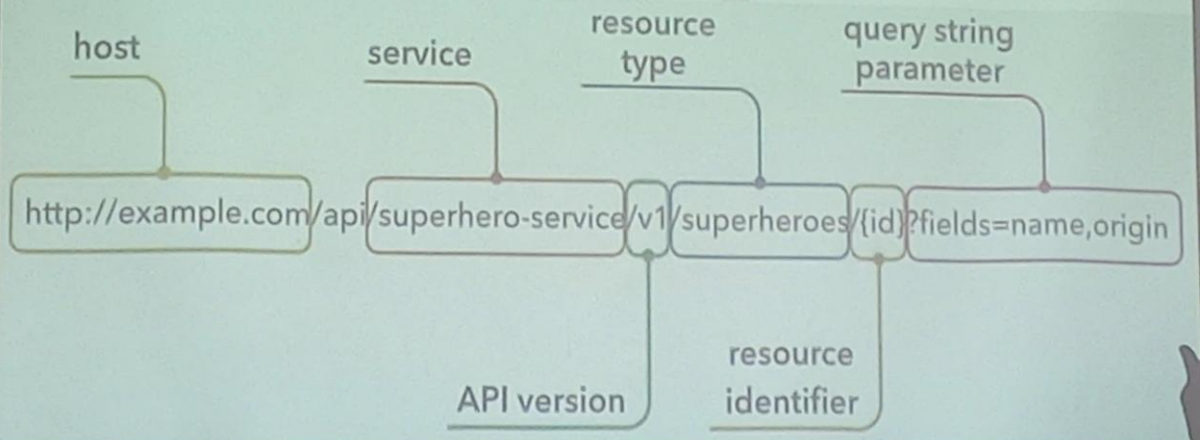
Example about REST APIs

- REST APIs can be called with all of the HTTP verbs. To get a resource, in this case, a user, a GET request is used. While the SOAP request holds the user's name in the body, a REST API accepts GET parameters from the URI.
- *GET https://restexample.com/users?name=John*
- As mentioned, REST APIs typically use the data format JSON. The user is represented in JSON like this:

```
{
  "name": "John",
  "age": 5,
  "address": "Greenville"
}
```

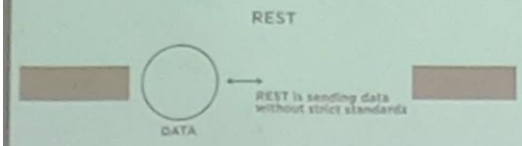

Anatomy of a REST API endpoint

- The main elements of a REST API endpoint: the host, the service, the API version, the resource type, the resource identifier, and any parameters.



REST

- REST is an acronym for Representational State Transfer and an architectural style for distributed hypermedia systems.
- REST that provides several standards and conventions that must be followed to facilitate communication between applications.
- REST is a software architectural style that defines the set of rules to be used for creating web services.
- Web services that follow the REST architectural style are known as REST API or RESTful web services. It allows requesting systems to access and manipulate web resources by using a uniform and predefined set of rules. Interaction in REST-based systems happens through the Internet's Hypertext Transfer Protocol (HTTP). It is one of the most widely used approaches for building web-based APIs.



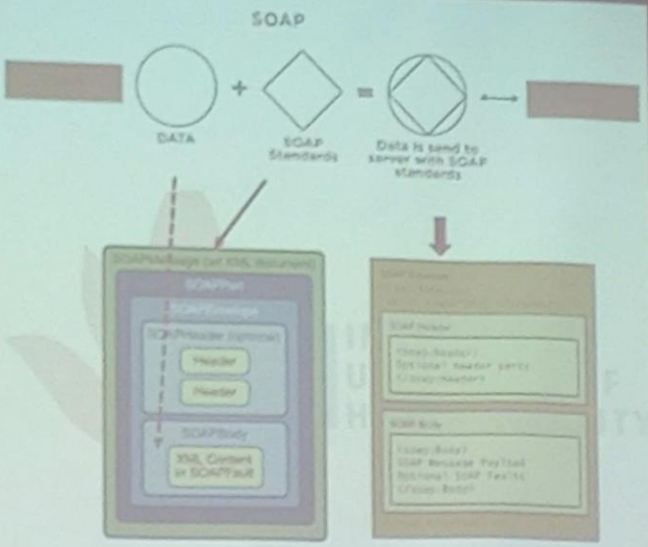
Six Guiding Principles of REST

01 Uniform Interface
02 Client-Server
03 Stateless
04 Cacheable
05 Layered System
06 Code on Demand (Optional)

24/08/2025 BY MR. HUYNH NA

SOAP APIs

- SOAP stands for Simple Object Access Protocol.
- This type of API is popular due to its strict structure and independence from any specific programming language.
- Examples of SOAP APIs include the following:
  - Salesforce SOAP API.
  - Workday SOAP API.
  - Sabre SOAP API.



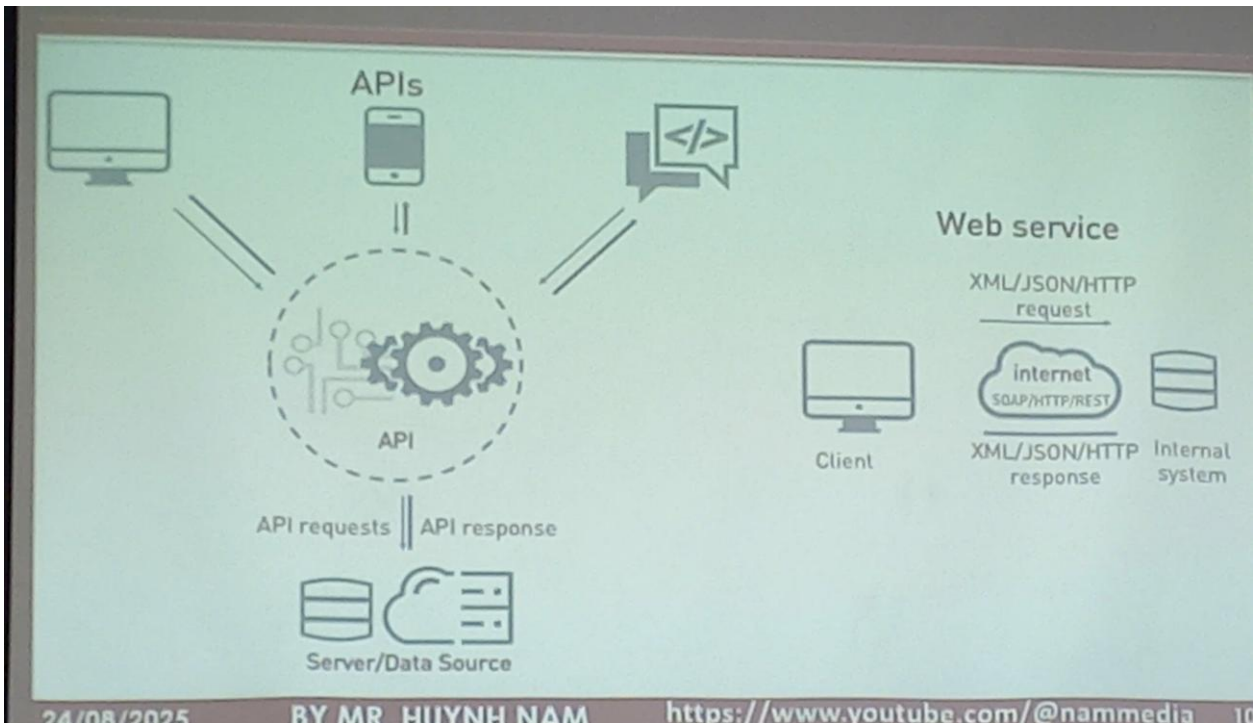
Example about SOAP

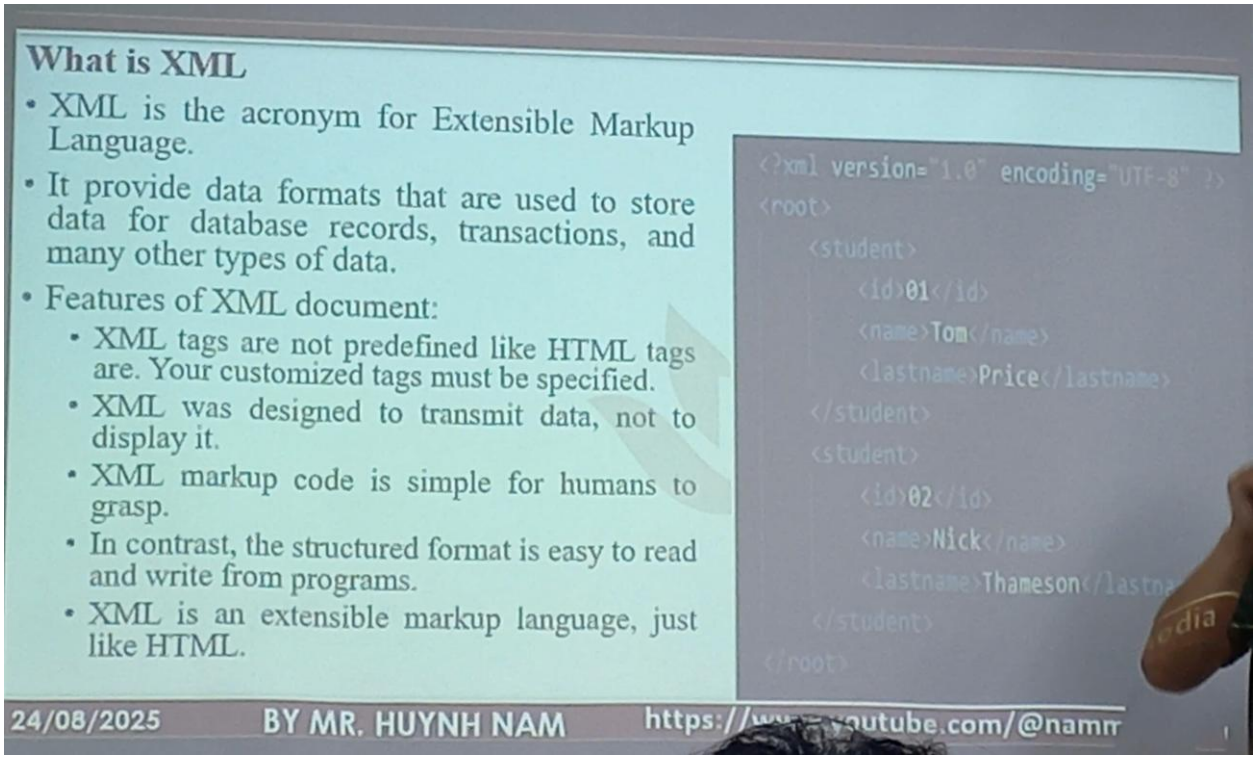
- Using SOAP, the request to the API is an HTTP POST request with an XML request body. The request body consists of an envelope which is a type of SOAP wrapper that identifies the requested API, and a SOAP body that holds the request parameters. In this case, we want to fetch the user with the name "John."

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:sch="http://www.soapexample.com/
xml/users">
  <soapenv:Header/>
  <soapenv:Body>
    <sch:UserDetailsRequest>
      <sch:name>John</sch:name>
    </sch:UserDetailsRequest>
  </soapenv:Body>
</soapenv:Envelope>
```


What is XML

- XML is the acronym for Extensible Markup Language.
- It provide data formats that are used to store data for database records, transactions, and many other types of data.
- Features of XML document:
  - XML tags are not predefined like HTML tags are. Your customized tags must be specified.
  - XML was designed to transmit data, not to display it.
  - XML markup code is simple for humans to grasp.
  - In contrast, the structured format is easy to read and write from programs.
  - XML is an extensible markup language, just like HTML.

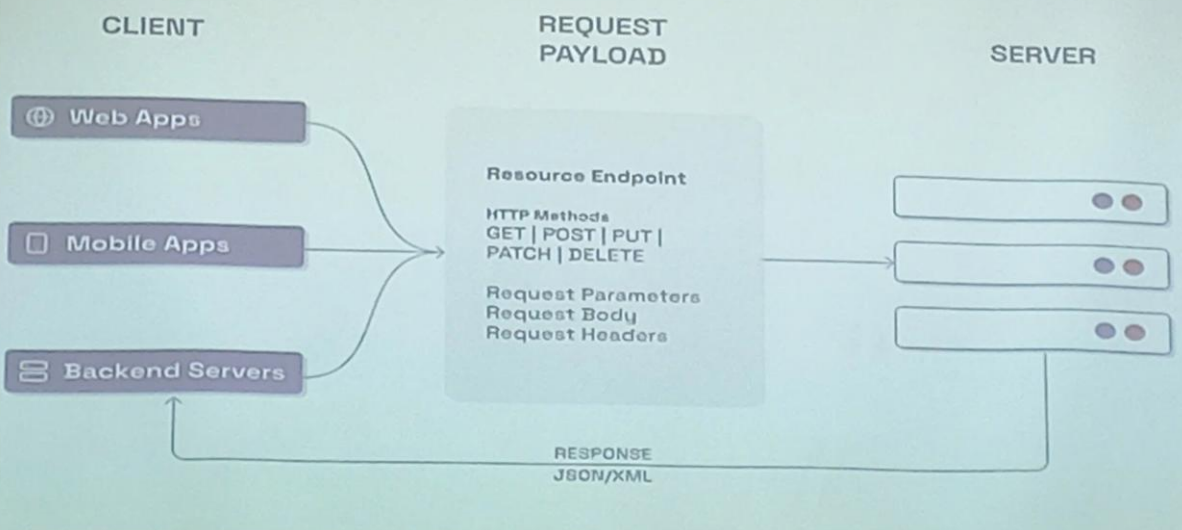

APIs
Web service
XML/JSON/HTTP
request
internet
SOAP/HTTP/REST
Client
XML/JSON/HTTP
response
Internal
system
API
API requests
API response
Server/Data Source
BY MR. HUYNH NAM
https://www.youtube.com/@nammedia


What is XML
• XML is the acronym for Extensible Markup Language.
• It provide data formats that are used to store data for database records, transactions, and many other types of data.
• Features of XML document:
• XML tags are not predefined like HTML tags are. Your customized tags must be specified.
• XML was designed to transmit data, not to display it.
• XML markup code is simple for humans to grasp.
• In contrast, the structured format is easy to read and write from programs.
• XML is an extensible markup language, just like HTML.
<?xml version="1.0" encoding="UTF-8" ?>
<root>
<student>
<id>01</id>
<name>Tom</name>
<lastname>Price</lastname>
</student>
<student>
<id>02</id>
<name>Nick</name>
<lastname>Thameson</lastname>
</student>
</root>
24/08/2025
BY MR. HUYNH NAM

Important Benefits of Using API

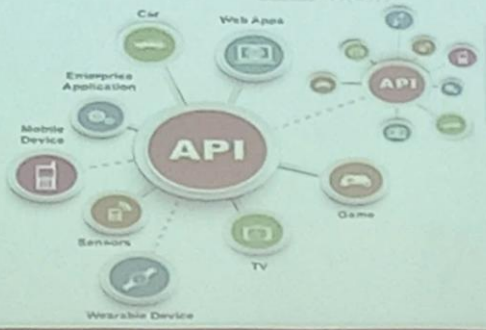
- Using API allows you to create a seamless stream of data transfer between applications and devices in real time.
  - Improve the quality of the user experience.
  - Speed of development and budget saving.
  - Complex operations become simple.
  - Scaling options.
- Based on API, things such as maps and routes, all kinds of mobile and desktop clients for social networks and other products that people use on a daily basis are built.

What does an API do



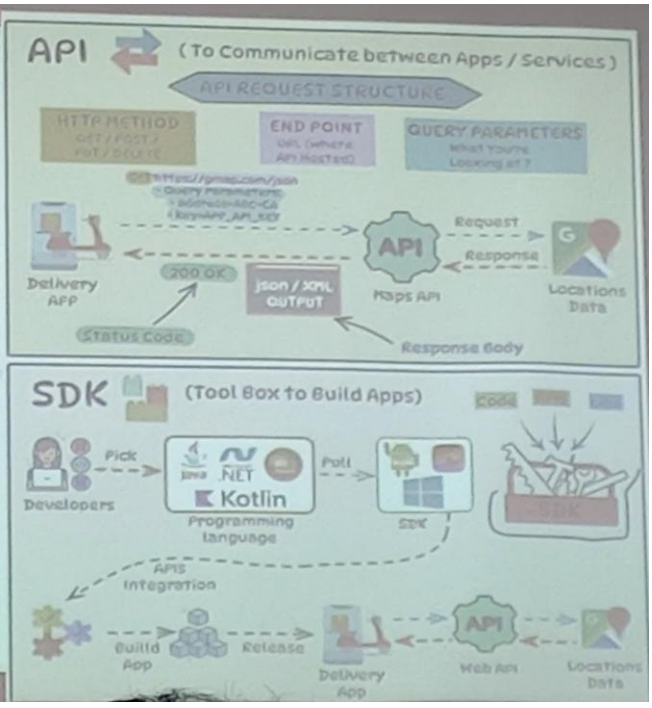
What is API

- API is the acronym for Application Programming Interface.
- A type of software interface that let two programs communicate with one another.
- An API define features that have nothing to do with the execution of each implementation
- APIs enable programmers to regularly reuse difficult, repetitive operations with only a little amount of code



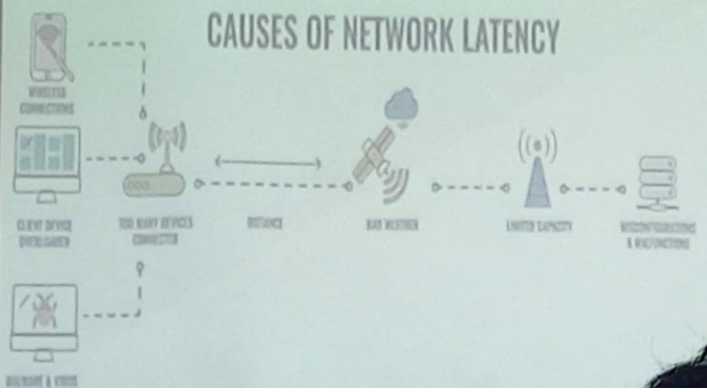
API vs SDK: Key Differences

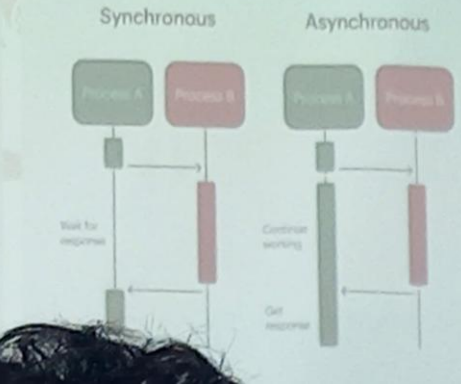
- **API**
  - An API is a set of rules and protocols that allows different software applications and services to communicate and share data.
- **SDK**
  - An SDK is a comprehensive package of tools, libraries, sample code, and documentation to simplify building apps on a specific platform, framework, or hardware.



Client-Server Communication

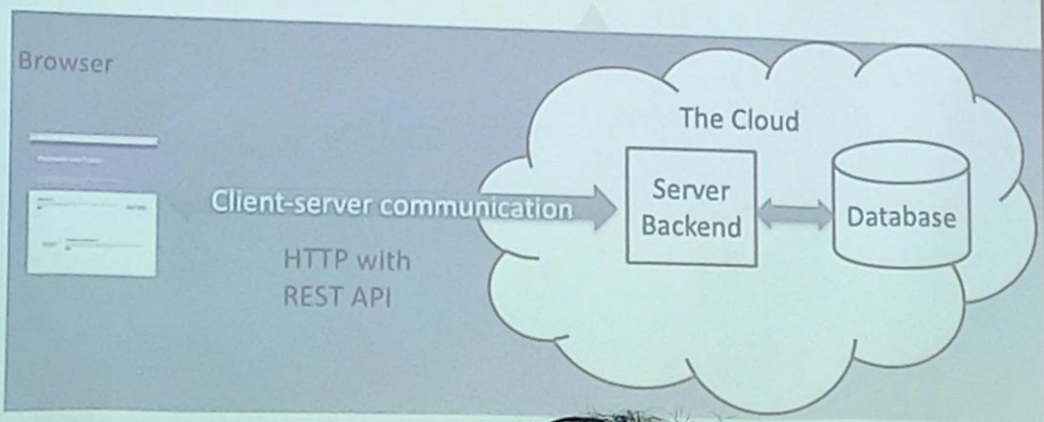
- Network operations cause unexpected delays
  - **Latency** refers to when a data pack is requested and the responses received. The requests could occur when users act on a network or web application.
- You need to write applications recognizing the asynchronous nature of communication
  - Data is not instantaneously available





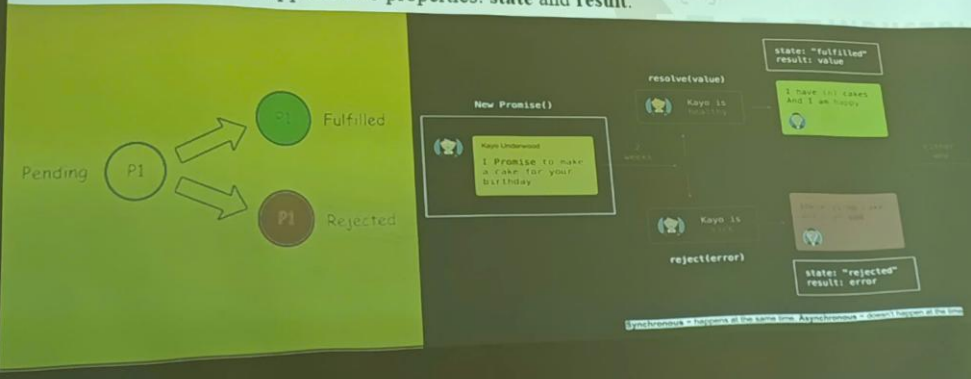
Client and Server

- Web applications are not stand-alone
- Many of them have a "Cloud" backend



What is Promise

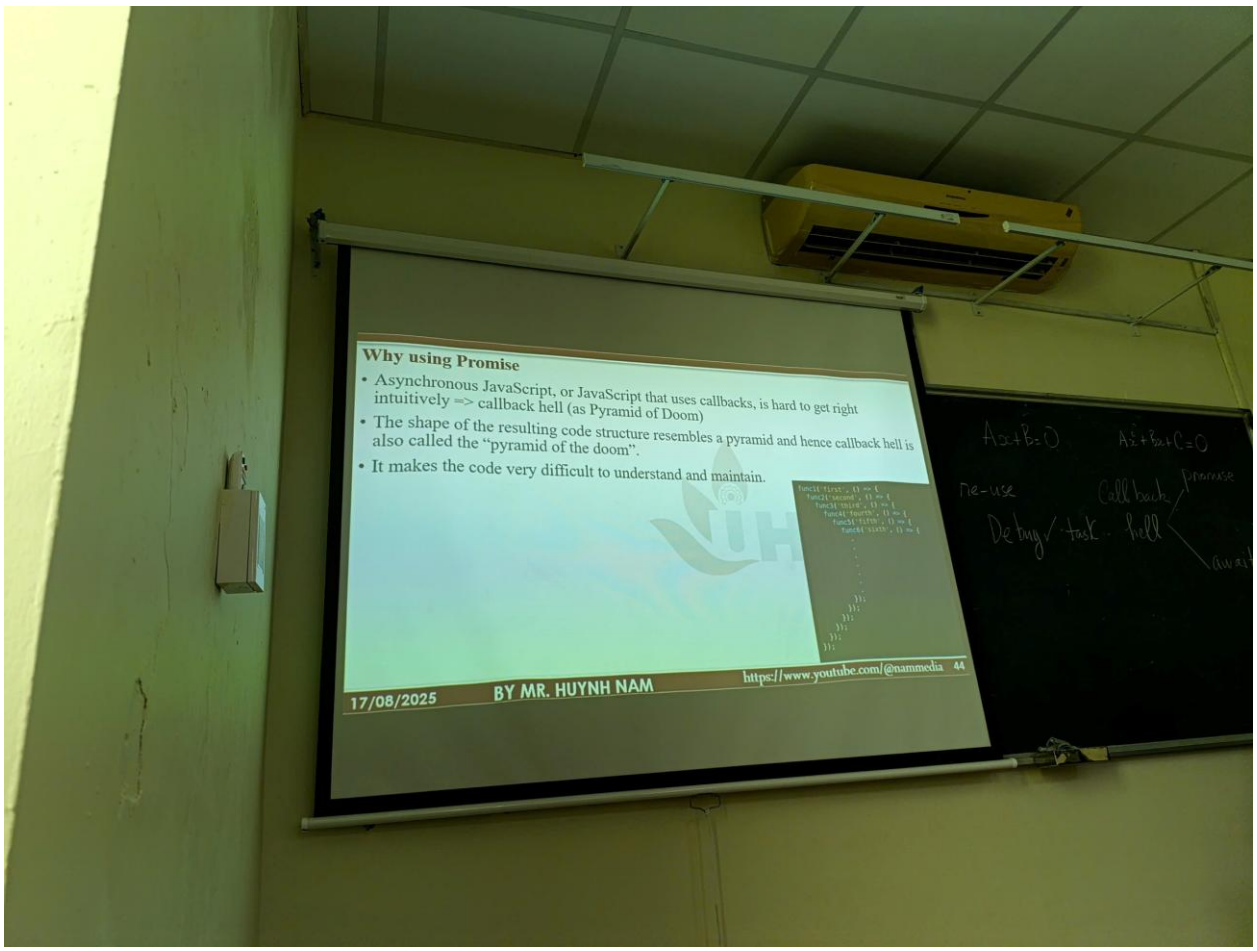
- A promise is an object with a syntactical coating introduced by ES6 which became another way of skipping writing callbacks to implement asynchronous operations.
- Web APIs like fetch() are implemented using the promising methodology.
- It makes the code more readable and is a solution to escape the callback hell.
- A JavaScript Promise object can be:
  - Fulfilled: the desired action has been resolved or completed successfully.
  - Pending : the desired action has neither been resolved nor been rejected and still in its initial state.
  - Rejected : the desired action has been rejected causing the desired operation to fail.
- The Promise object supports two properties: **state** and **result**.

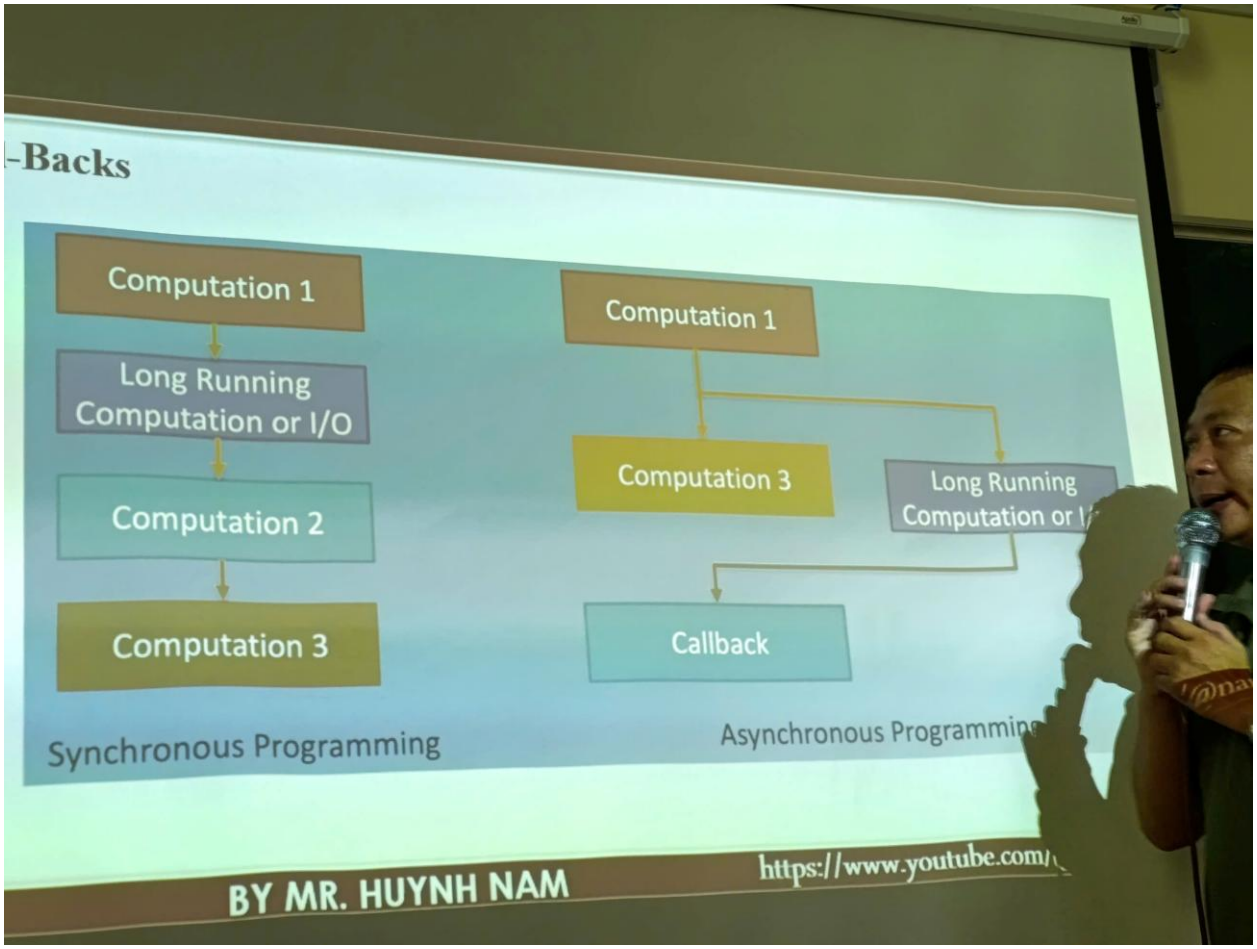


Async/Await

- This is the second method to escape the callback hell.
- It serves as a better way of approaching promises
- It internally resolves promises as well as makes it easier for developers to write readable code.
- The **async** function is made to stop its execution until a Promise() is either fulfilled or rejected. The function will resume only once the promise is settled. After it resumes, the value of the set of await expression will contain the value of the settled Promise().
- **await** expression shows the rejected result if the Promise() gets rejected.

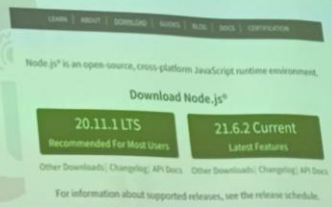
17/08/2025 BY MR. HUYNH NAM https://www.youtube.com/@nammedia 47


Why using Promise
• Asynchronous JavaScript, or JavaScript that uses callbacks, is hard to get right intuitively => callback hell (as Pyramid of Doom)
• The shape of the resulting code structure resembles a pyramid and hence callback hell is also called the "pyramid of the doom".
• It makes the code very difficult to understand and maintain.
17/08/2025
BY MR. HUYNH NAM
https://www.youtube.com/@nammedia 44


-Backs
Computation 1
Long Running Computation or I/O
Computation 2
Computation 3
Synchronous Programming
Computation 1
Computation 3
Long Running Computation or I/
Callback
Asynchronous Programming
BY MR. HUYNH NAM
https://www.youtube.com/

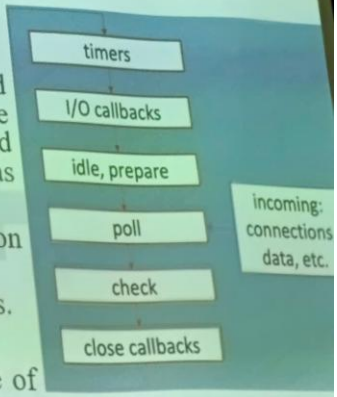
Environment Setup

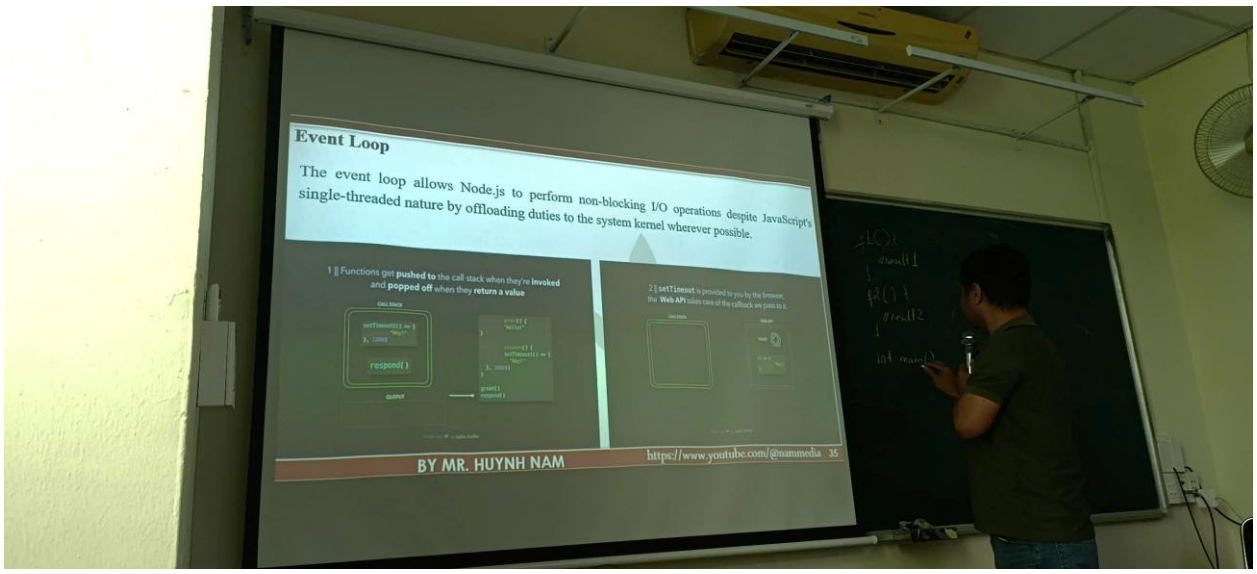
- Link Download and Install NodeJS: https://nodejs.org/
- Node Package Manager (NPM) offers two primary features:
  - Node.js packages and modules' online repositories, searchable via search.nodejs.org.
  - A command-line tool for managing Node.js package dependencies, version control, and installation.
  - A package contains:
    - JS files
    - package.json (manifest)
  - npm –version
- package.json
  - is used to specify a package's properties
- Initializing package.json
  - To initialize a package.json file for your project, type at the prompt in your project directory:
- npm init
  - Follow along and answer the prompts to initialize



Event Loop and Emitters

- Node.js offers events and callbacks to let users accomplish this. The event loop is an infinite loop that, after searching the event queue for any triggered events, executes those that are discovered and deletes them from the queue.
- Phases in Event Loop
  1. Timers: The setTimeout() and setInterval()-scheduled callbacks are executed at this stage. Timer callbacks are created by the setTimeout() and setInterval() functions, and they attempt to run as soon as the specified time has eclapsed
  2. Pending Callbacks: The deferred callbacks' execution happens at this stage.
  3. Poll: It executes I/O callbacks and fetches fresh I/O events.
  4. Check: Here, setImmediate() callbacks are triggered.
  5. Close callbacks: Socket.on ('close'), for example, is one of the close callbacks.




Event Loop
The event loop allows Node.js to perform non-blocking I/O operations despite JavaScript's single-threaded nature by offloading duties to the system kernel wherever possible.
BY MR. HUYNH NAM
https://www.youtube.com/@nammedia 35


Node Architecture
Node Core / Standard Library (JS)
Node Bindings (C++)
Chrome V8 (C++)
libuv (C)
17/08/2025
BY MR. HUYNH NAM

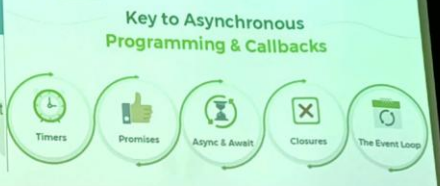
Node Package Manager (NPM)

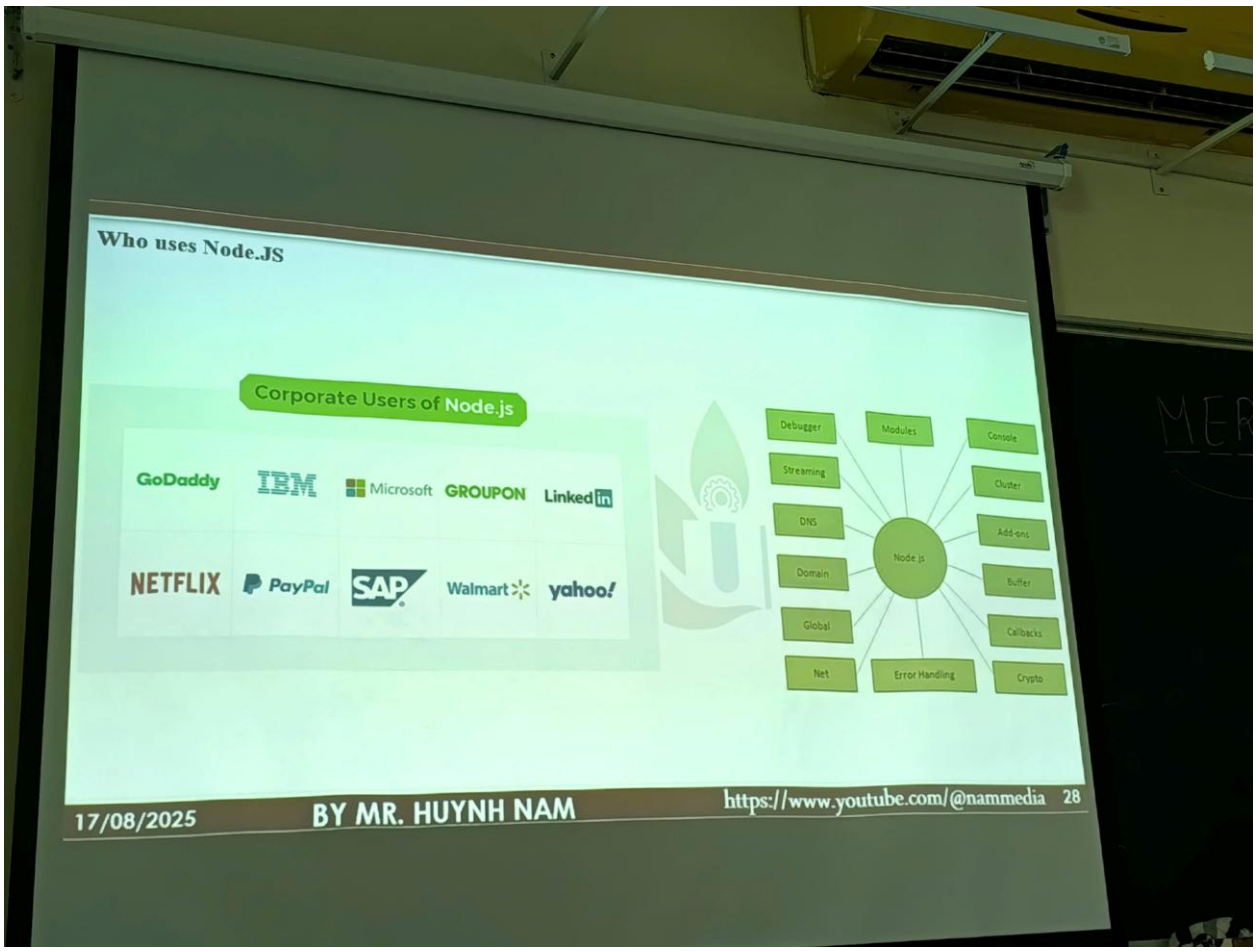
- Node package manager (NPM): manages ecosystem of node modules / packages
- A package contains:
  - JS files
  - package.json (manifest)
    - It serves as documentation for what packages your project depends on.
    - It allows you to specify the versions of a package that your project can use using semantic versioning rules.
    - Makes your build reproducible, which means that its way easier to share with other developers.

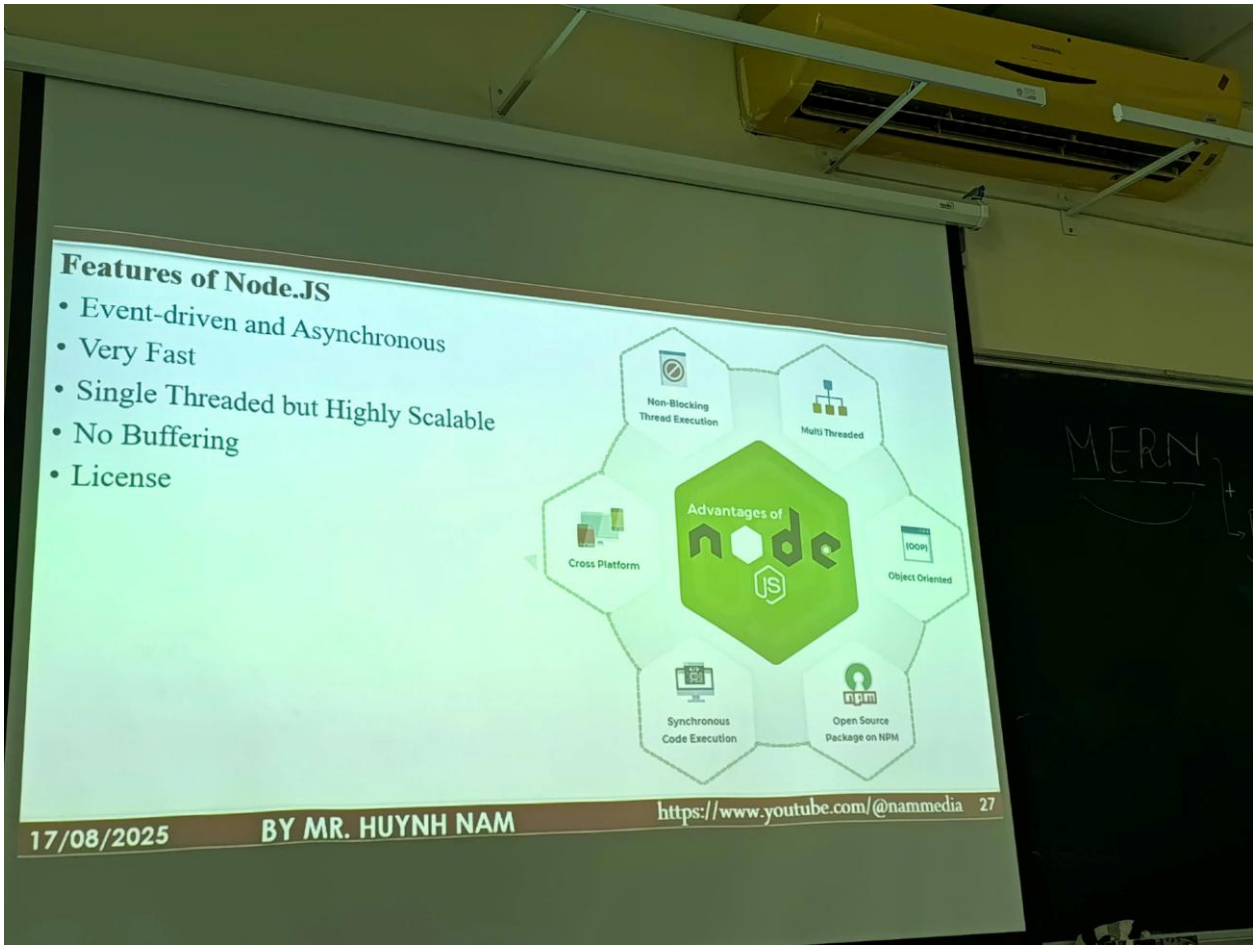
Why Node.JS

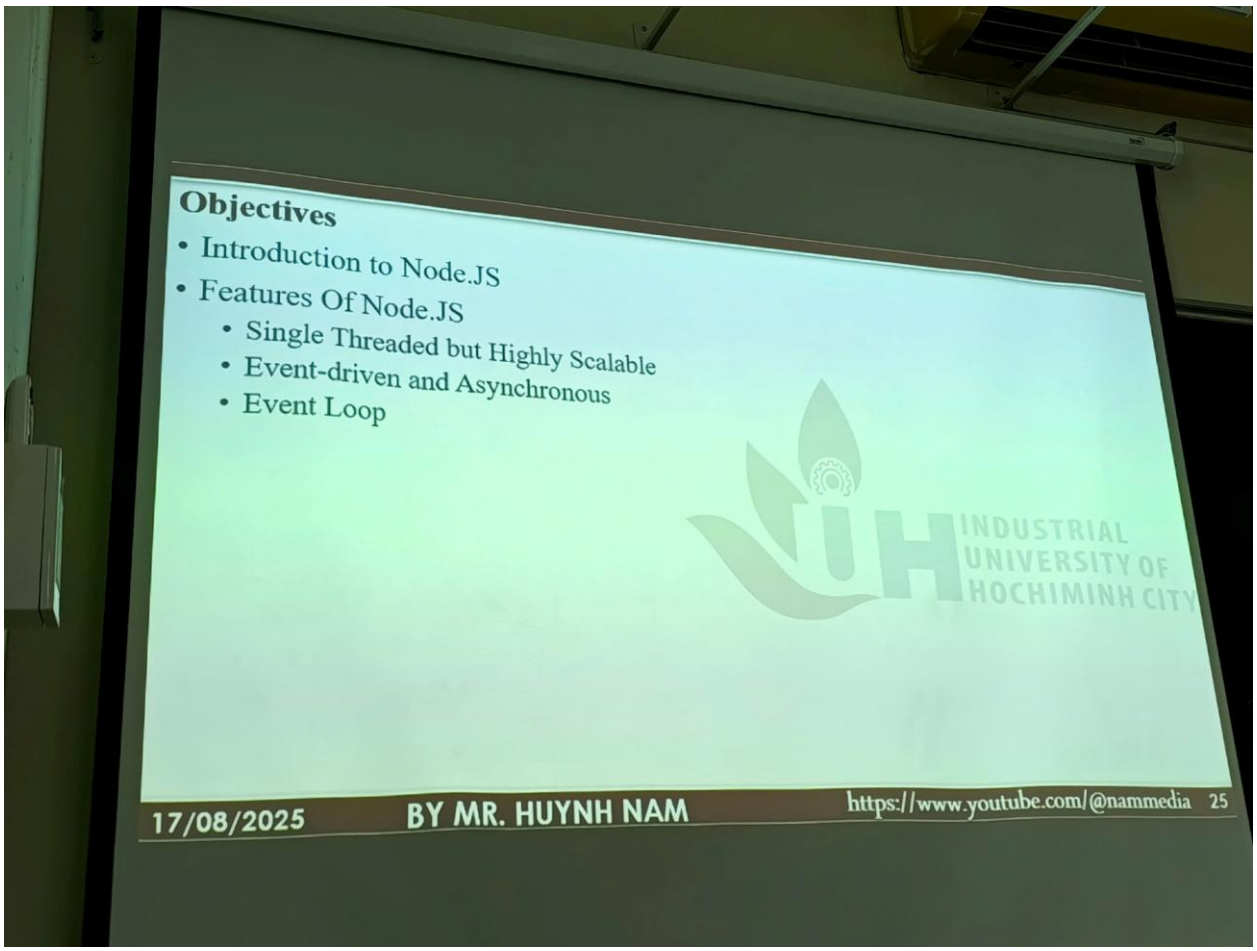
- Node.js eliminates the waiting and simply continues with the next request.
- Node.js runs single-threaded, non-blocking, asynchronous programming, which is very memory efficient.

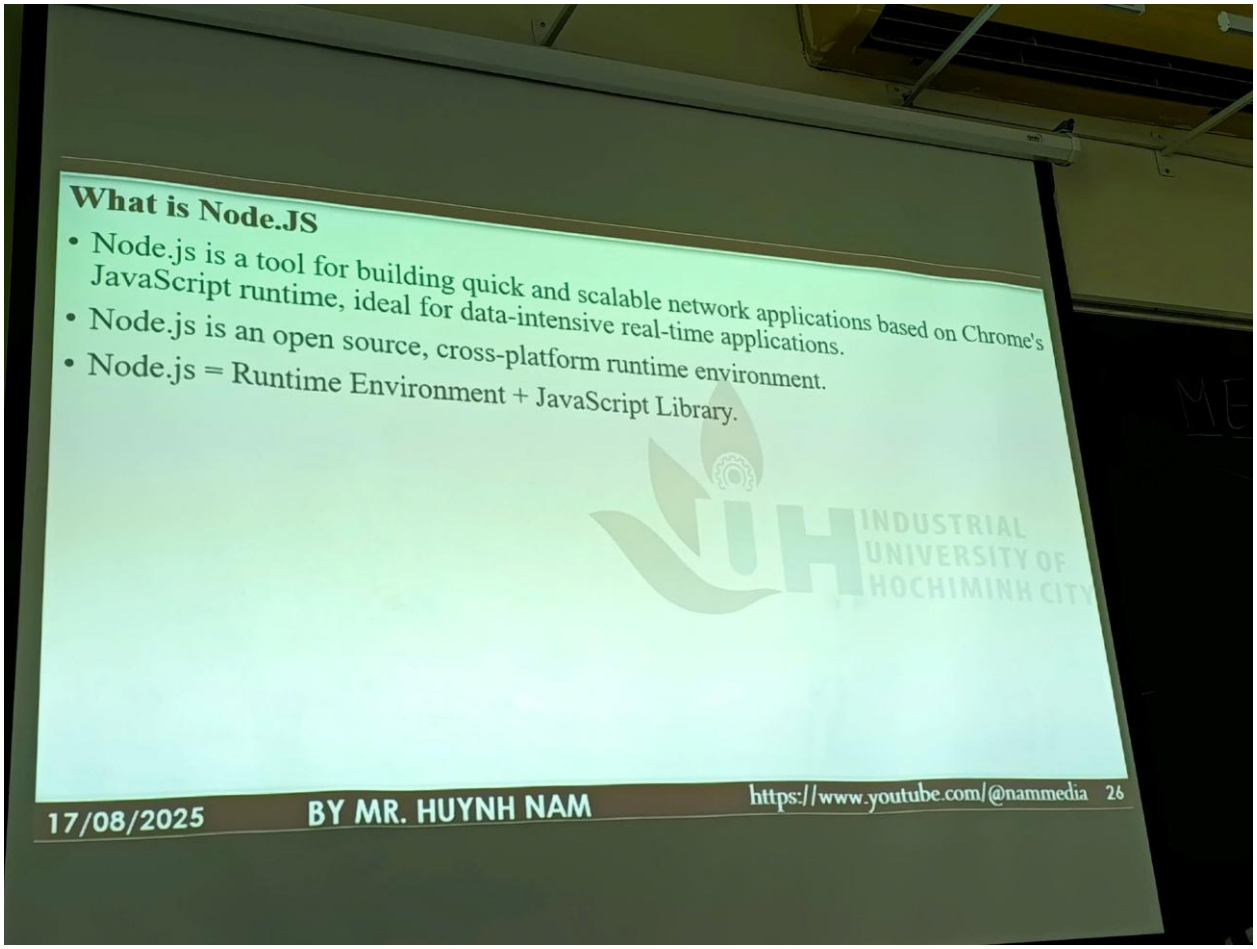
How traditional server handles a file request	How Nodejs handles a file request
1. Transmits the task to the file system on the computer 2. Wait until the opens and begins reading it. 3. Gives the client their content back. 4. Prepared to tackle the next request	1. Transmits the task to the file system on the computer. 2. Prepared to manage the next request. 3. The server sends the content to the client after the file system has opened and read the file.

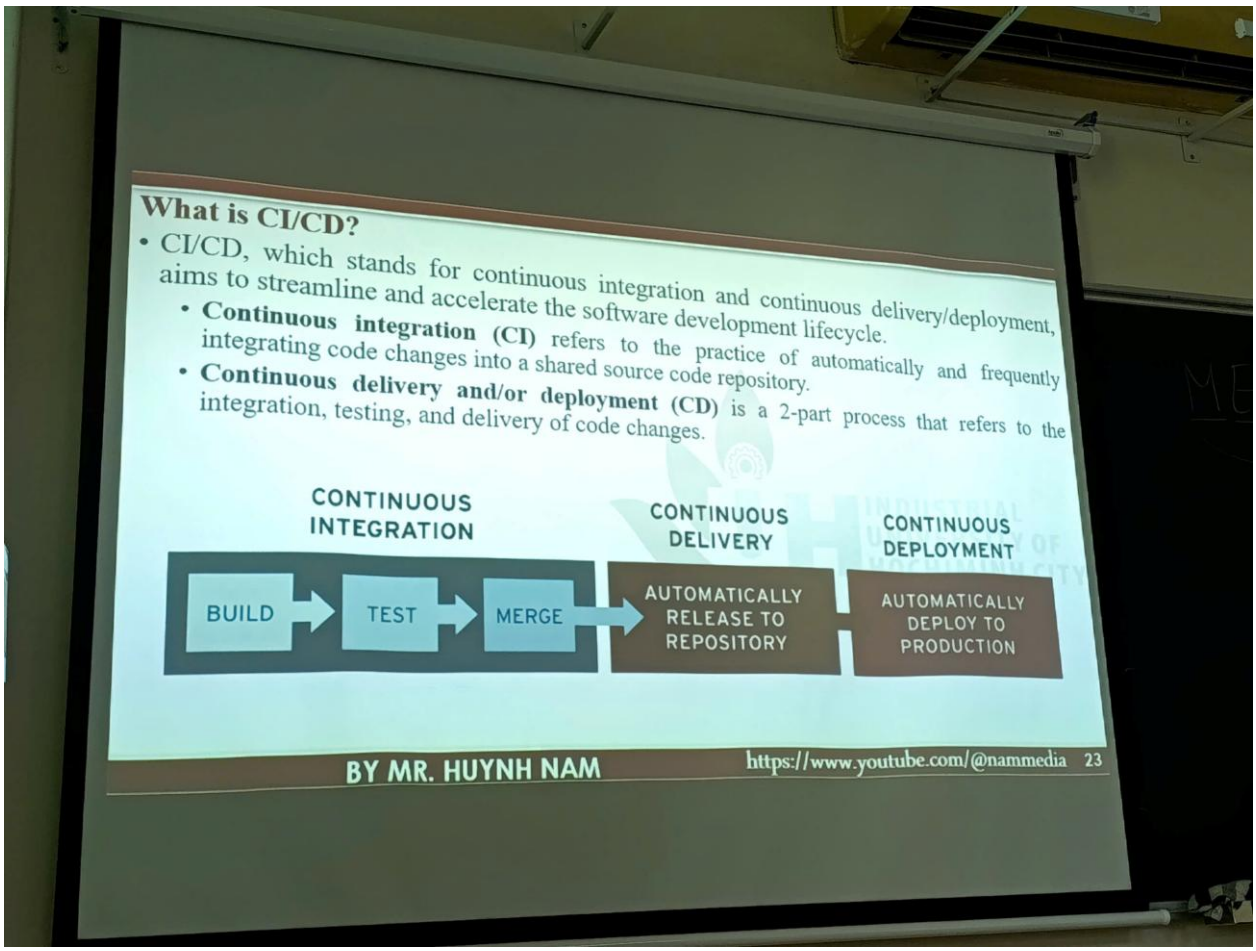


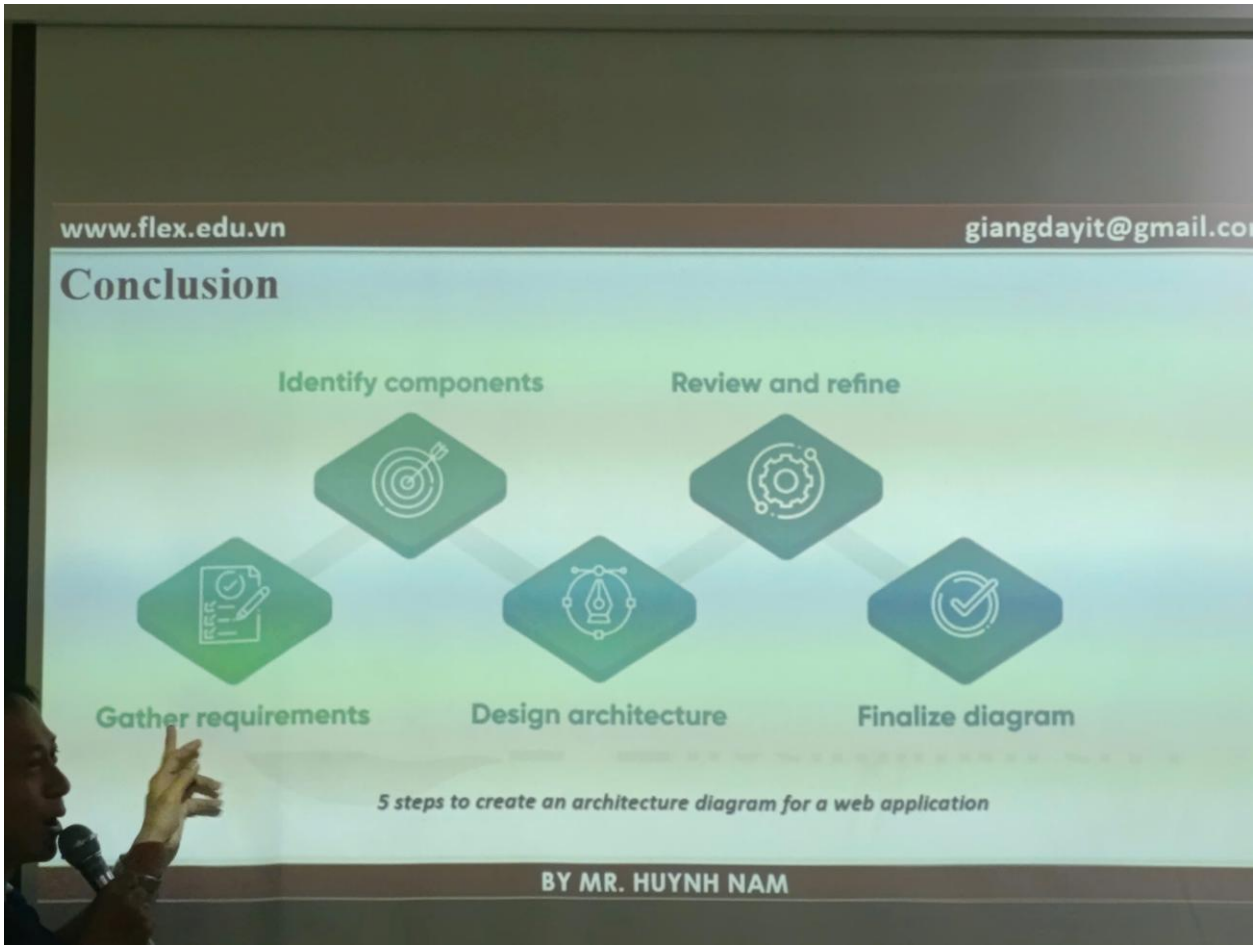

Who uses Node.JS
Corporate Users of Node.js
GoDaddy
IBM
Microsoft
GROUPON
LinkedIn
NETFLIX
PayPal
SAP
Walmart
yahoo!
Debugger
Modules
Console
Streaming
Cluster
DNS
Add-ons
Node.js
Domain
Buffer
Global
Callbacks
Net
Error Handling
Crypto
17/08/2025
BY MR. HUYNH NAM
https://www.youtube.com/@nammedia
28

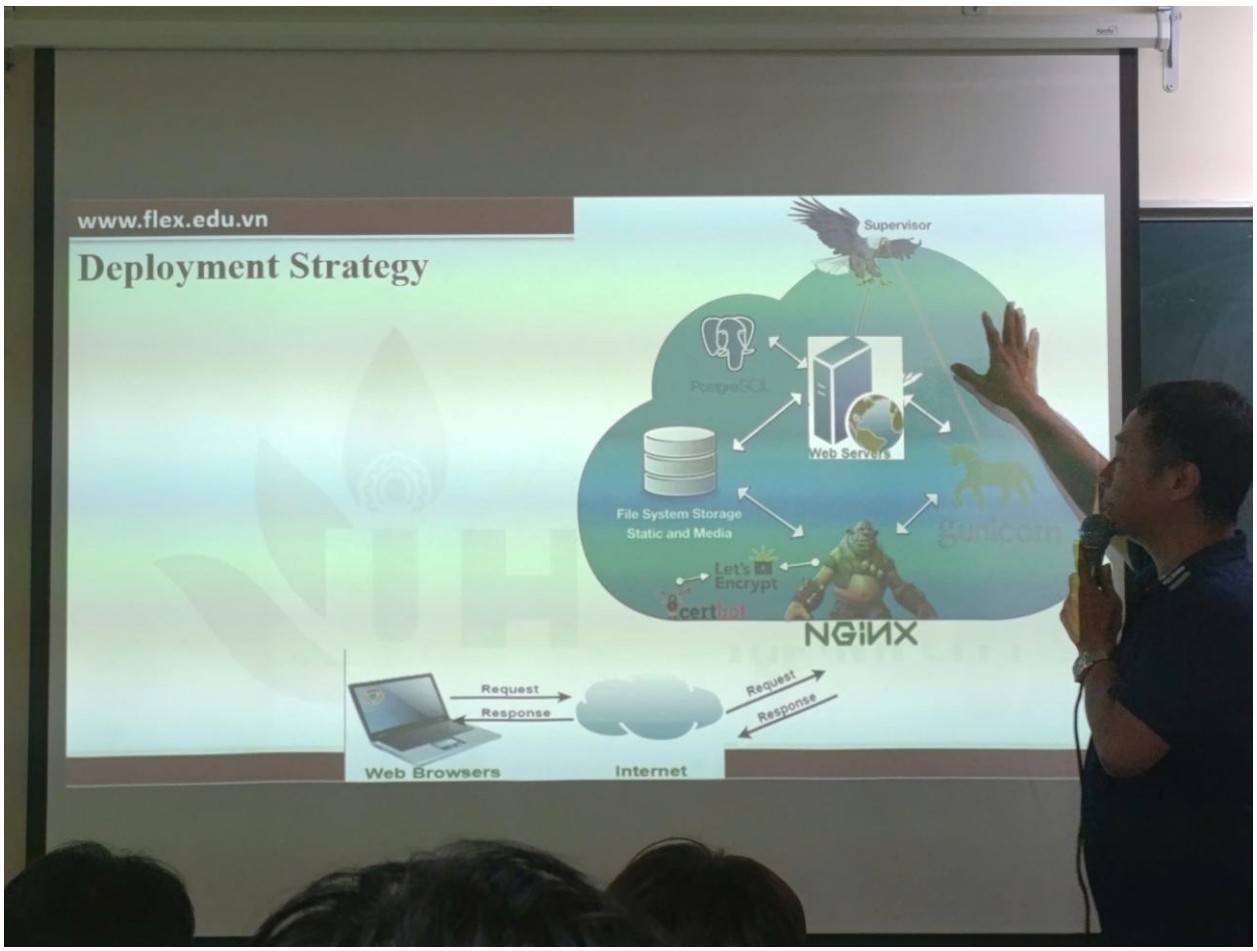

Features of Node.JS
• Event-driven and Asynchronous
• Very Fast
• Single Threaded but Highly Scalable
• No Buffering
• License
Non-Blocking Thread Execution
Multi Threaded
Advantages of node
Cross Platform
Object Oriented
Synchronous Code Execution
Open Source Package on NPM
17/08/2025
BY MR. HUYNH NAM
https://www.youtube.com/@nammedia
27

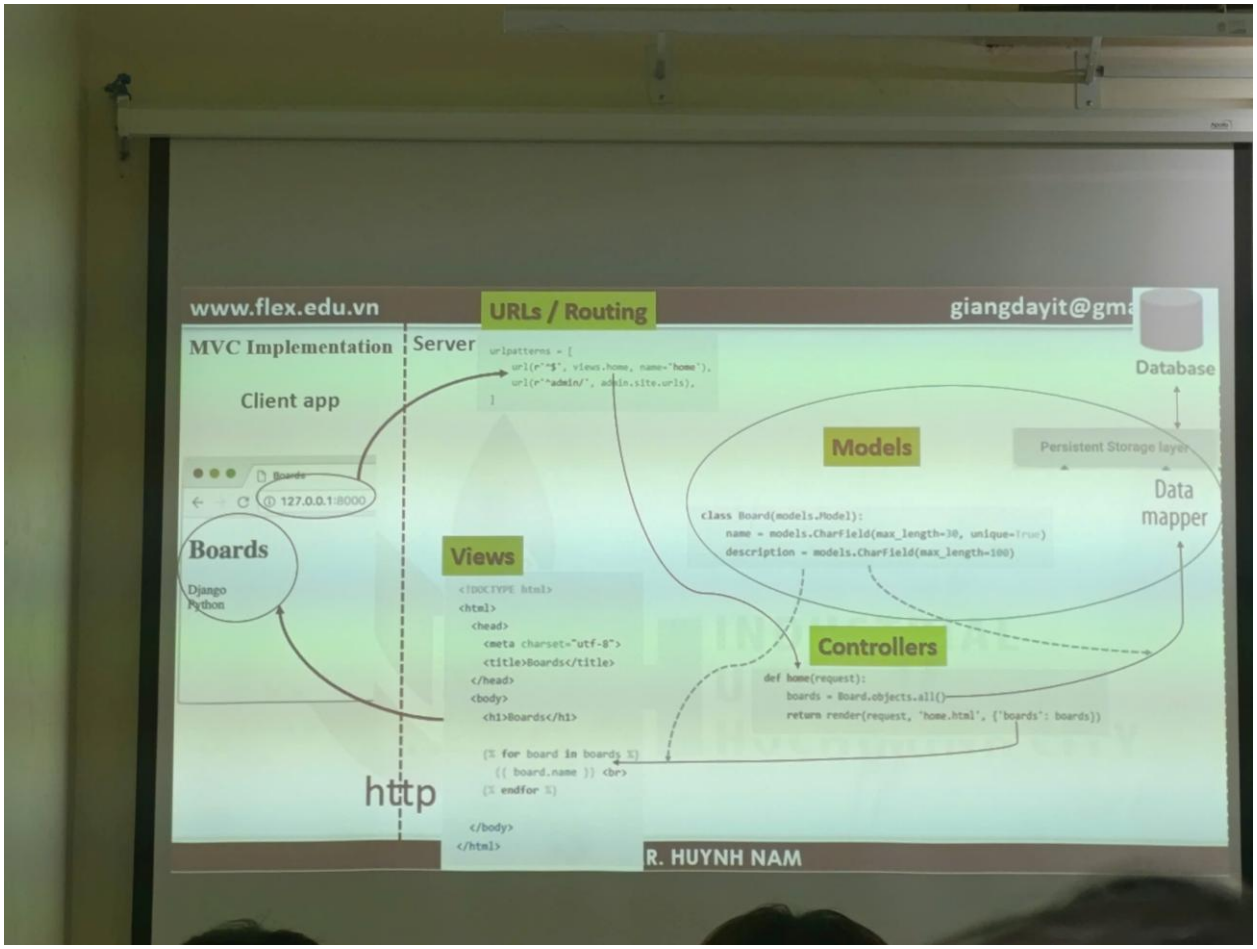

Objectives
• Introduction to Node.JS
• Features Of Node.JS
• Single Threaded but Highly Scalable
• Event-driven and Asynchronous
• Event Loop
INDUSTRIAL UNIVERSITY OF HOCHIMINH CITY
17/08/2025
BY MR. HUYNH NAM
https://www.youtube.com/@nammedia 25

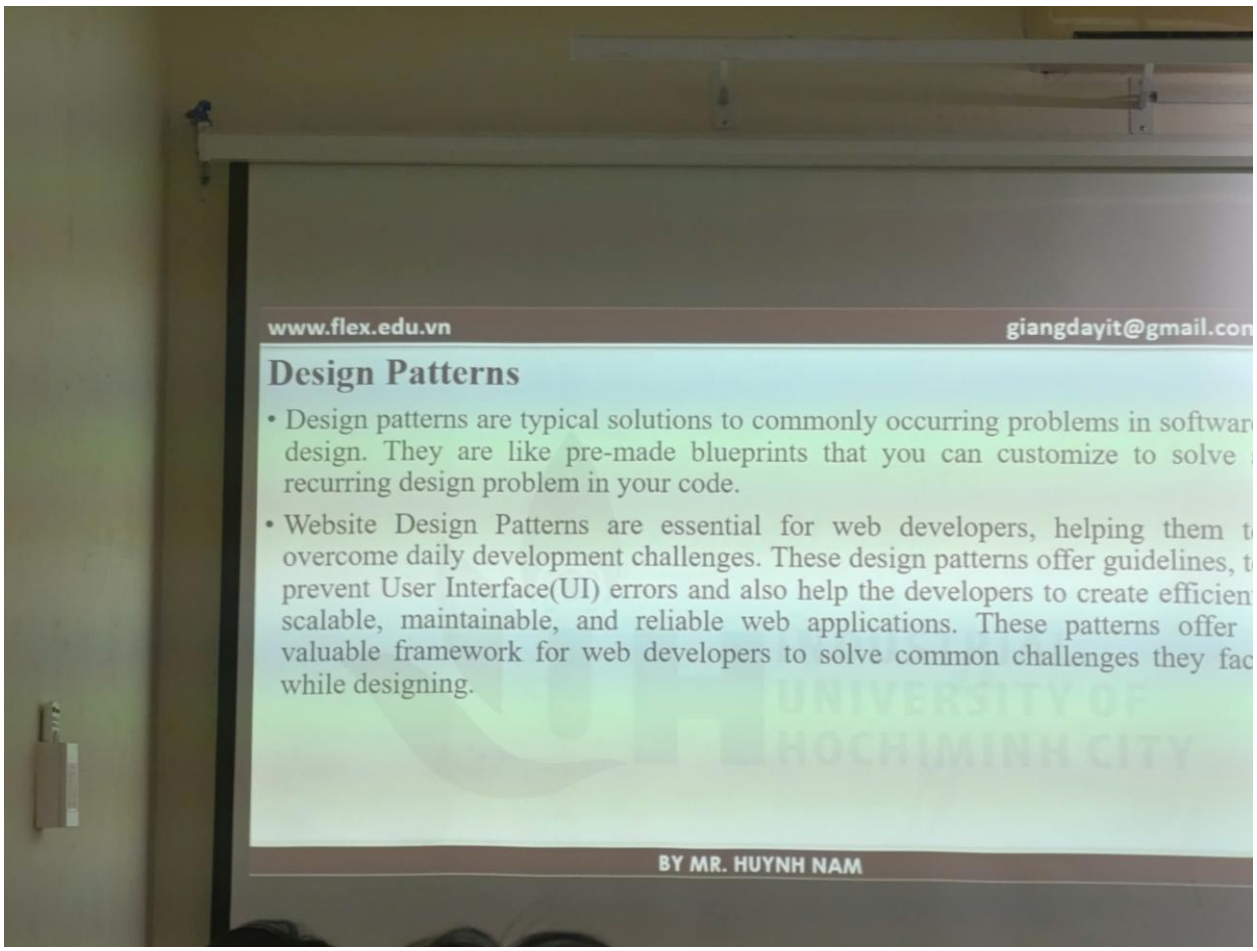

What is Node.JS
• Node.js is a tool for building quick and scalable network applications based on Chrome's JavaScript runtime, ideal for data-intensive real-time applications.
• Node.js is an open source, cross-platform runtime environment.
• Node.js = Runtime Environment + JavaScript Library.
INDUSTRIAL UNIVERSITY OF HOCHIMINH CITY
17/08/2025
BY MR. HUYNH NAM
https://www.youtube.com/@nammedia 26

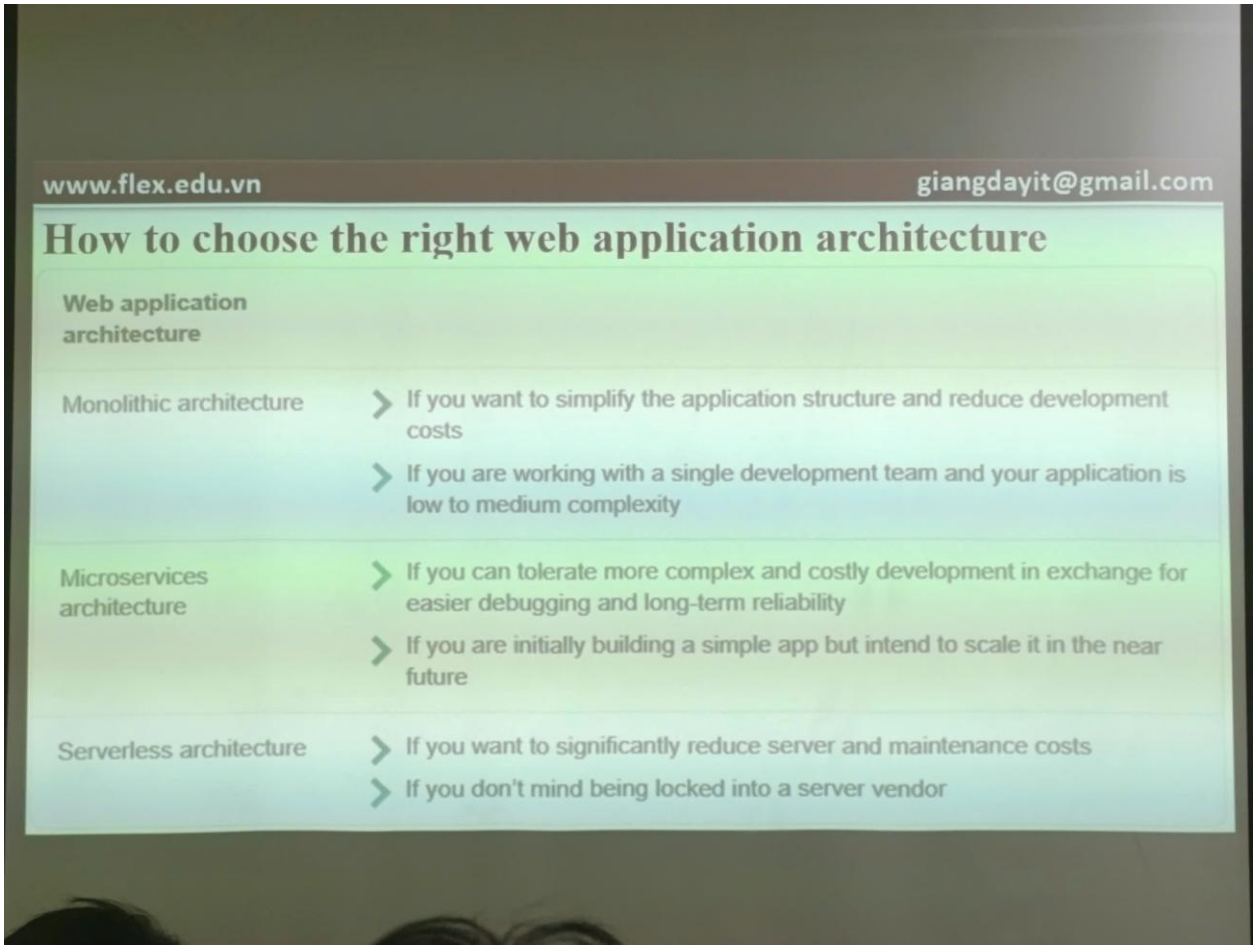

What is CI/CD?
• CI/CD, which stands for continuous integration and continuous delivery/deployment, aims to streamline and accelerate the software development lifecycle.
• Continuous integration (CI) refers to the practice of automatically and frequently integrating code changes into a shared source code repository.
• Continuous delivery and/or deployment (CD) is a 2-part process that refers to the integration, testing, and delivery of code changes.
CONTINUOUS INTEGRATION
CONTINUOUS DELIVERY
CONTINUOUS DEPLOYMENT
BUILD
TEST
MERGE
AUTOMATICALLY RELEASE TO REPOSITORY
AUTOMATICALLY DEPLOY TO PRODUCTION
BY MR. HUYNH NAM
https://www.youtube.com/@nammedia
23

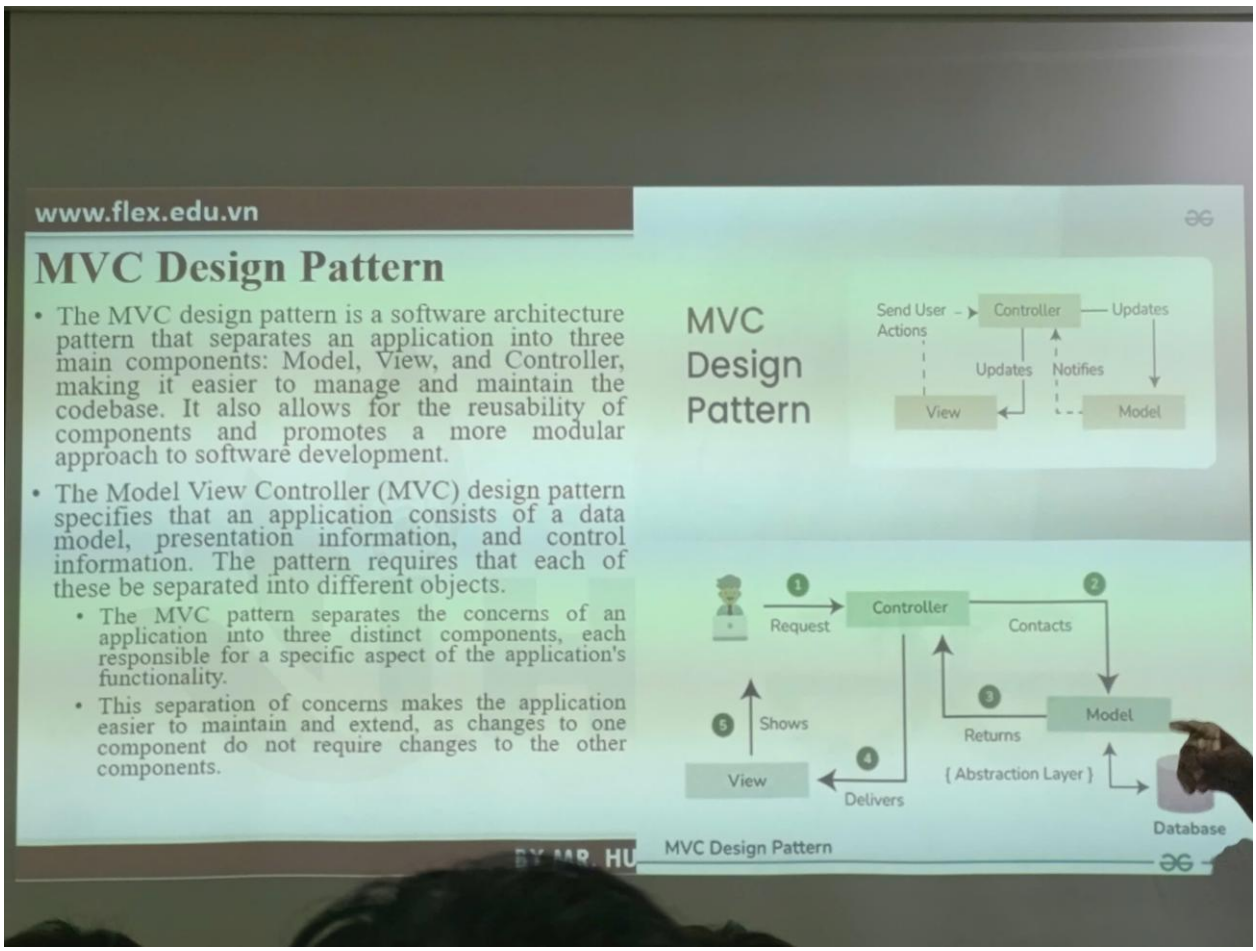

www.flex.edu.vn
giangdayit@gmail.co
Conclusion
Identify components
Review and refine
Gather requirements
Design architecture
Finalize diagram
5 steps to create an architecture diagram for a web application
BY MR. HUYNH NAM

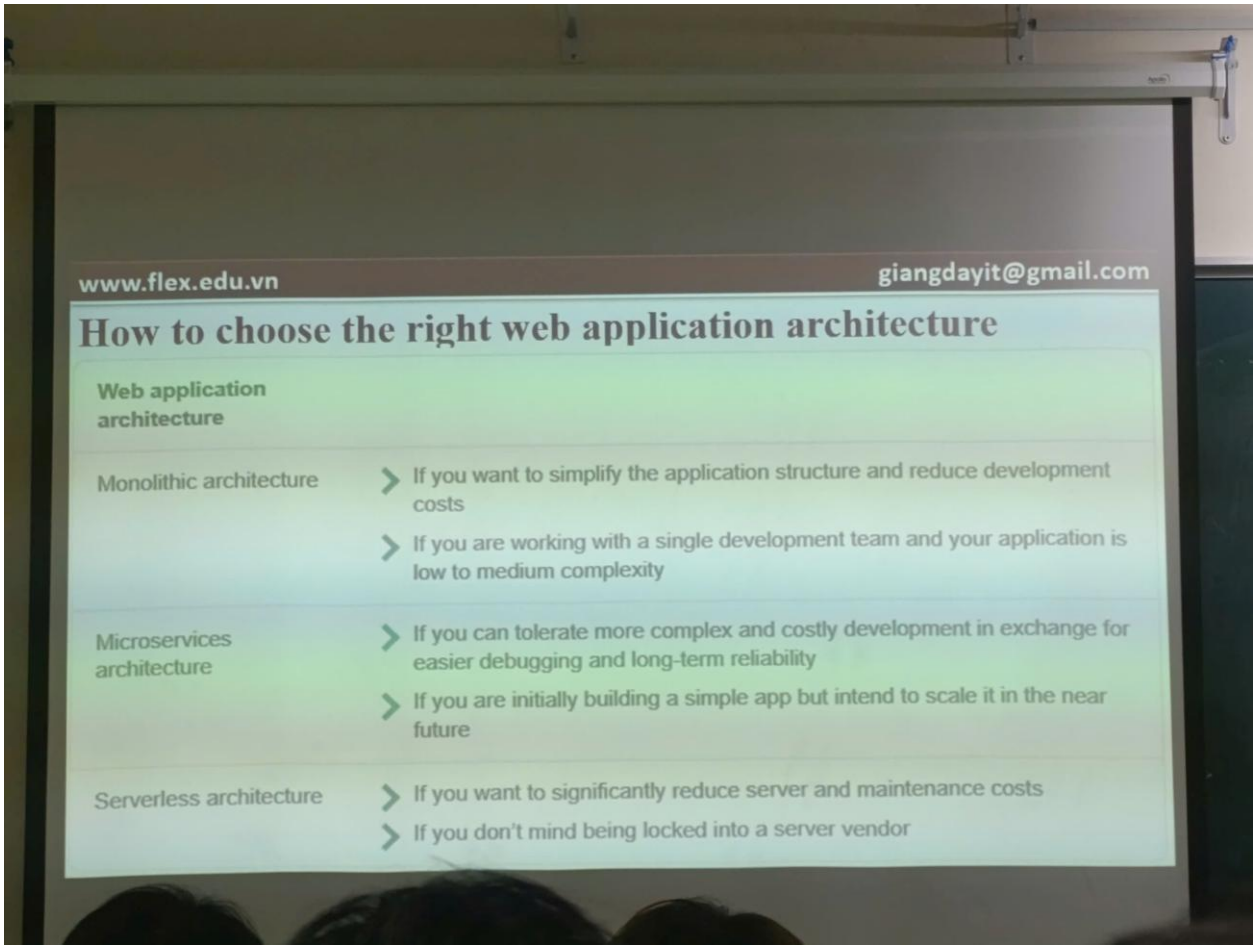

www.flex.edu.vn
Deployment Strategy
Supervisor
PostgreSQL
Web Servers
File System Storage
Static and Media
Gunicorn
Let's Encrypt
certbot
NGINX
Request
Response
Request
Response
Web Browsers
Internet


www.flex.edu.vn
URLs / Routing
giangdayit@gma
MVC Implementation
Server
Client app
Database
Models
Persistent Storage layer
Data mapper
Boards
Django
Python
Views
Controllers
http
R. HUYNH NAM


www.flex.edu.vn
giangdayit@gmail.com
Design Patterns
• Design patterns are typical solutions to commonly occurring problems in softwar design. They are like pre-made blueprints that you can customize to solve recurring design problem in your code.
• Website Design Patterns are essential for web developers, helping them t overcome daily development challenges. These design patterns offer guidelines, t prevent User Interface(UI) errors and also help the developers to create efficien scalable, maintainable, and reliable web applications. These patterns offer valuable framework for web developers to solve common challenges they fac while designing.
BY MR. HUYNH NAM


www.flex.edu.vn
giangdayit@gmail.com
How to choose the right web application architecture
Web application architecture
Monolithic architecture
If you want to simplify the application structure and reduce development costs
If you are working with a single development team and your application is low to medium complexity
Microservices architecture
If you can tolerate more complex and costly development in exchange for easier debugging and long-term reliability
If you are initially building a simple app but intend to scale it in the near future
Serverless architecture
If you want to significantly reduce server and maintenance costs
If you don't mind being locked into a server vendor

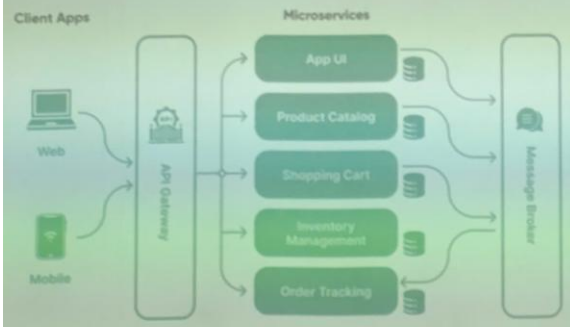

www.flex.edu.vn
MVC Design Pattern
• The MVC design pattern is a software architecture pattern that separates an application into three main components: Model, View, and Controller, making it easier to manage and maintain the codebase. It also allows for the reusability of components and promotes a more modular approach to software development.
• The Model View Controller (MVC) design pattern specifies that an application consists of a data model, presentation information, and control information. The pattern requires that each of these be separated into different objects.
• The MVC pattern separates the concerns of an application into three distinct components, each responsible for a specific aspect of the application's functionality.
• This separation of concerns makes the application easier to maintain and extend, as changes to one component do not require changes to the other components.
MVC Design Pattern
Send User Actions
Controller
Updates
Updates
Notifies
View
Model
Request
Controller
Contacts
Shows
Returns
Model
View
Delivers
{ Abstraction Layer }
Database
MVC Design Pattern


www.flex.edu.vn
giangdayit@gmail.com
How to choose the right web application architecture
Web application architecture
Monolithic architecture
If you want to simplify the application structure and reduce development costs
If you are working with a single development team and your application is low to medium complexity
Microservices architecture
If you can tolerate more complex and costly development in exchange for easier debugging and long-term reliability
If you are initially building a simple app but intend to scale it in the near future
Serverless architecture
If you want to significantly reduce server and maintenance costs
If you don't mind being locked into a server vendor

www.flex.edu.vn giangdayit@gmail.com

Microservices web application architecture

- Microservices architecture is a modern, web application architecture that breaks down a large, complex application into smaller, independent services. These services communicate with each other using APIs and are independently deployable.



Pros:

- Small, focused teams that can work independently
- Highly scalable and resilient
- Allows for the use of different technologies for different services
- Fault isolation and better resilience

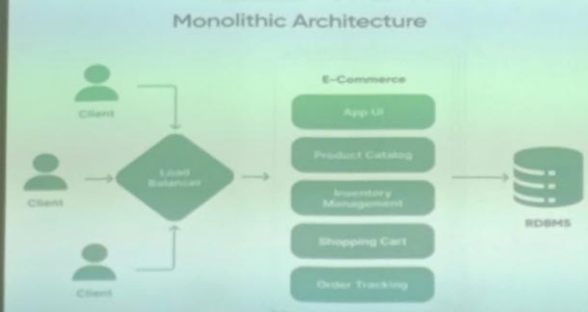
Cons:

- Increased complexity in system design
- Network problems and latency
- Difficulties in developing and testing
- Potential for data integrity issues due to multiple databases.

www.flex.edu.vn giangdayit@gmail.com

Monolithic web application architecture

- The most basic, traditional model: monolithic architecture. This web application architecture relies on a single, unified codebase for all web application components. This means that the codebase contains all of the application's code, including the user interface, business logic, and database access logic. Additionally, all components of the application share the same runtime environment.

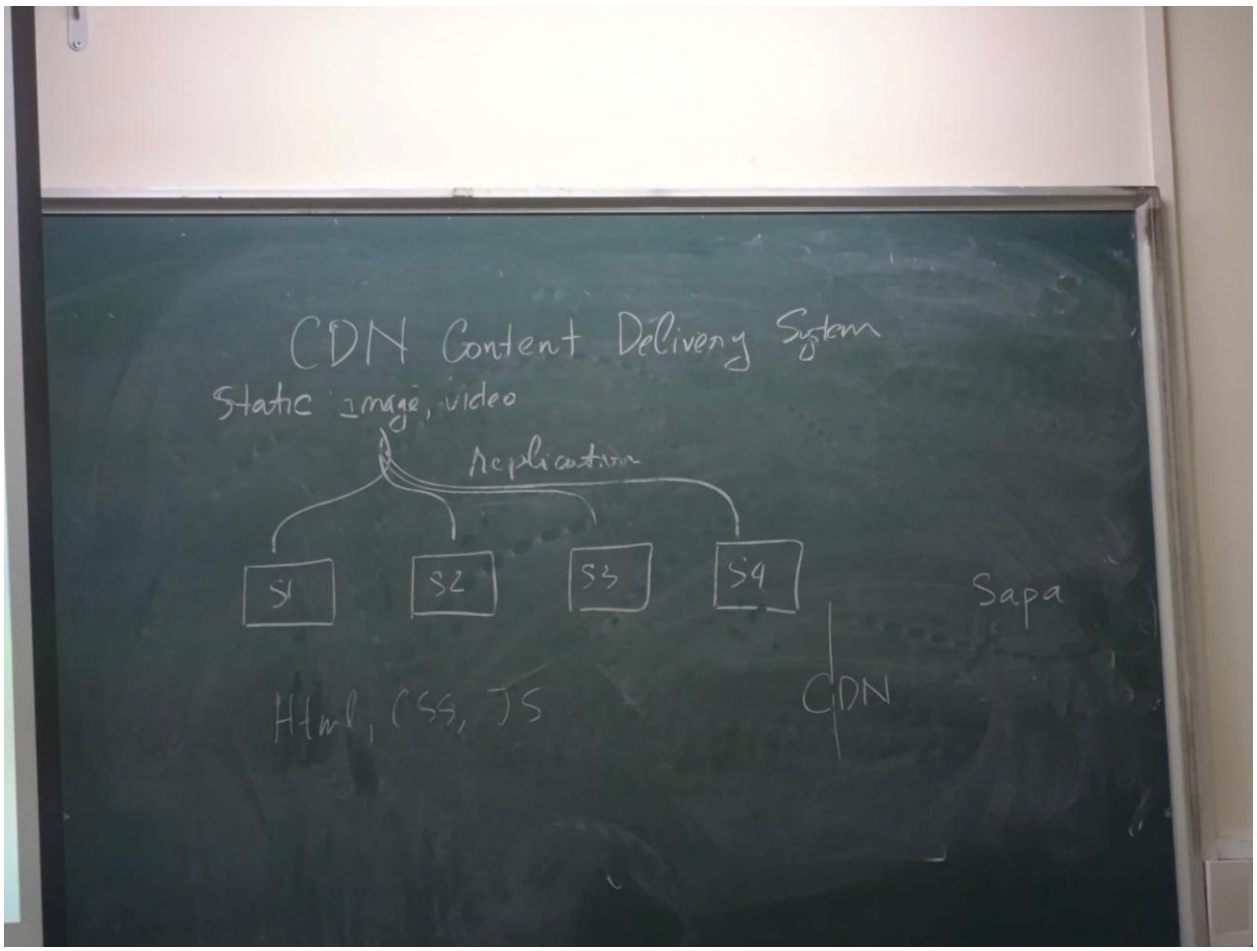


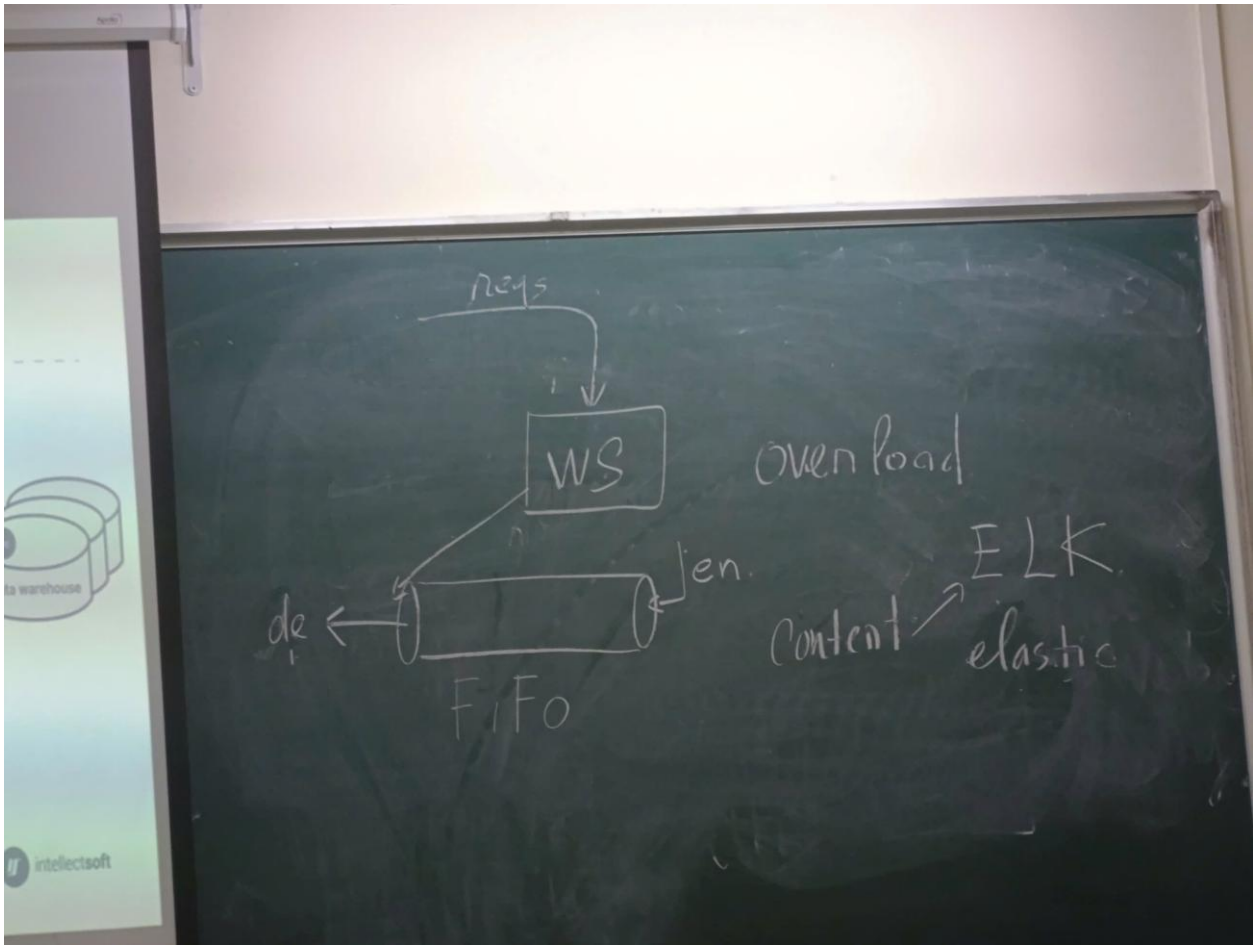
Pros

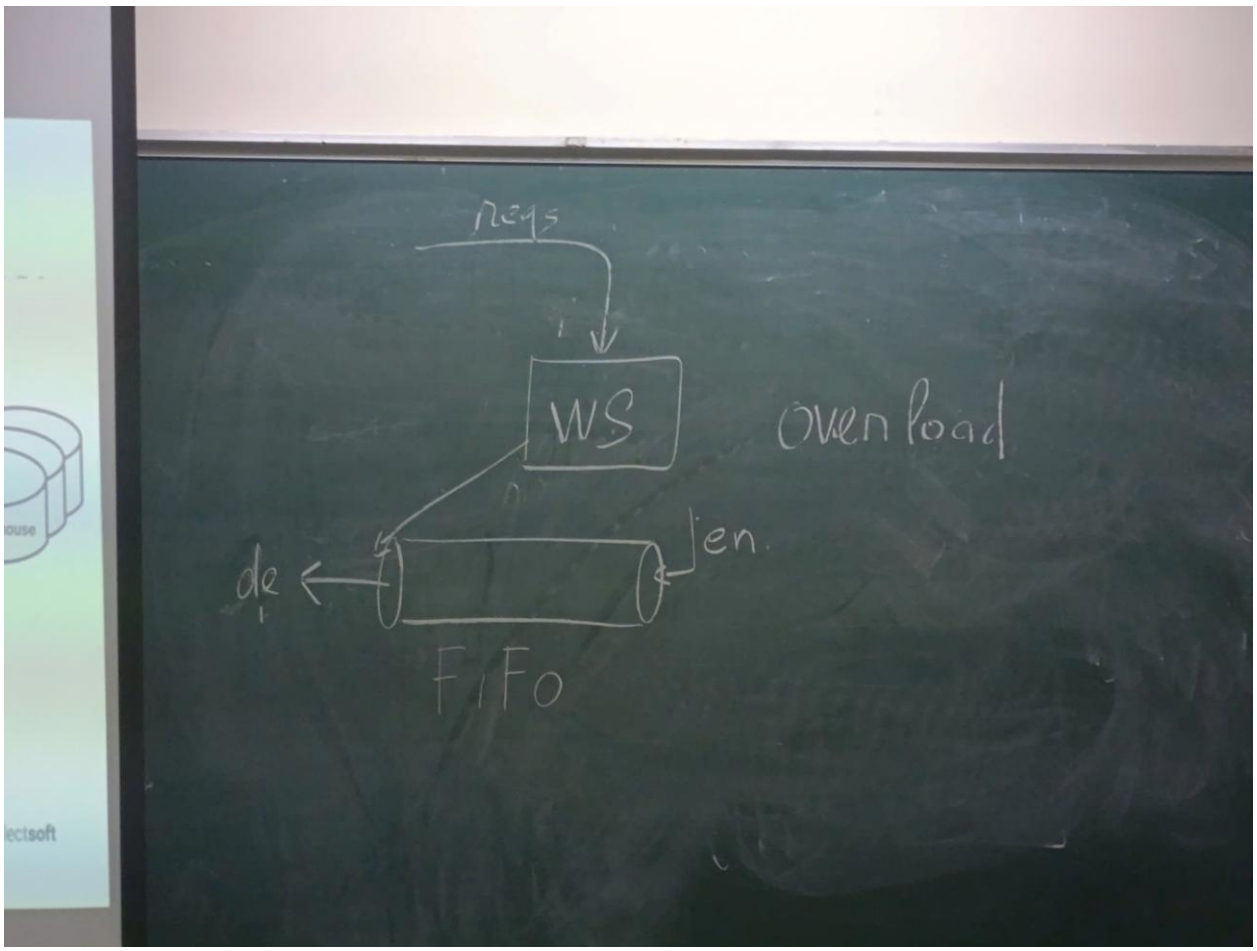
- Single codebase makes development and maintenance easier.
- Low complexity due to fewer moving parts.
- Easier to debug and deploy code.
- Improved performance due to fewer network requests.

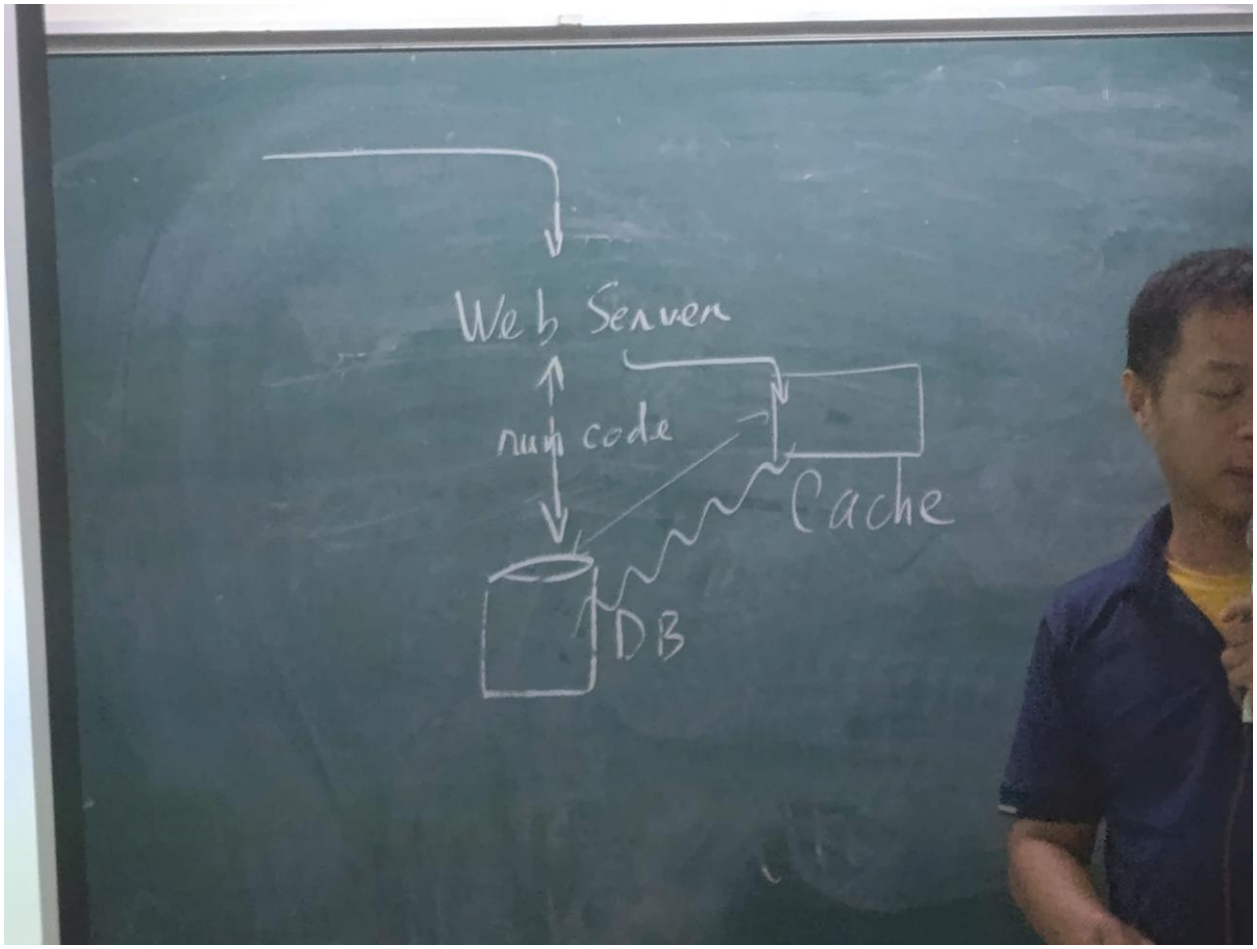
Cons

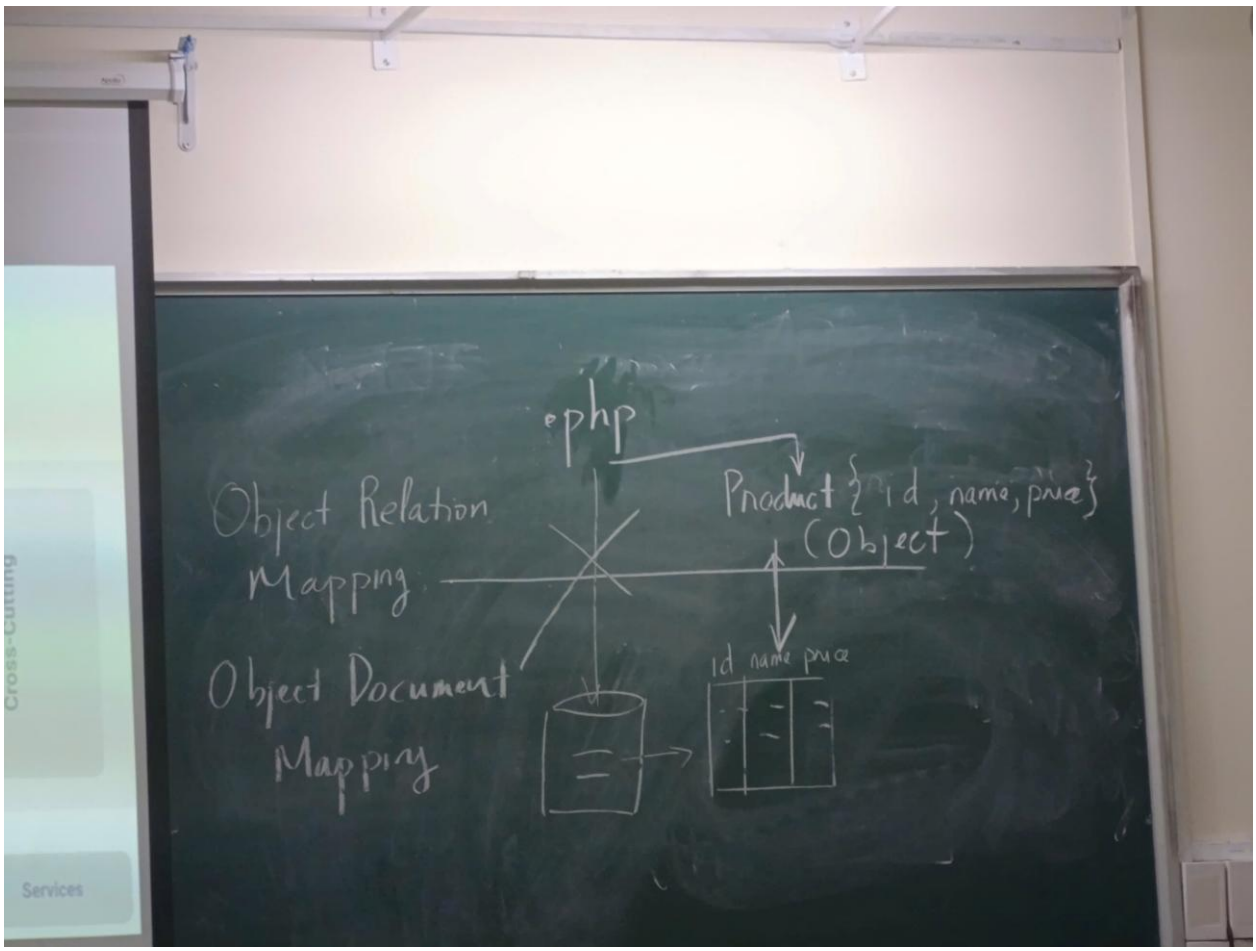
- Difficult to scale individual components.
- Not well-suited for rapid development or frequent changes.
- Lack of modularity can make it difficult to reuse code.
- Not designed to handle large traffic loads.

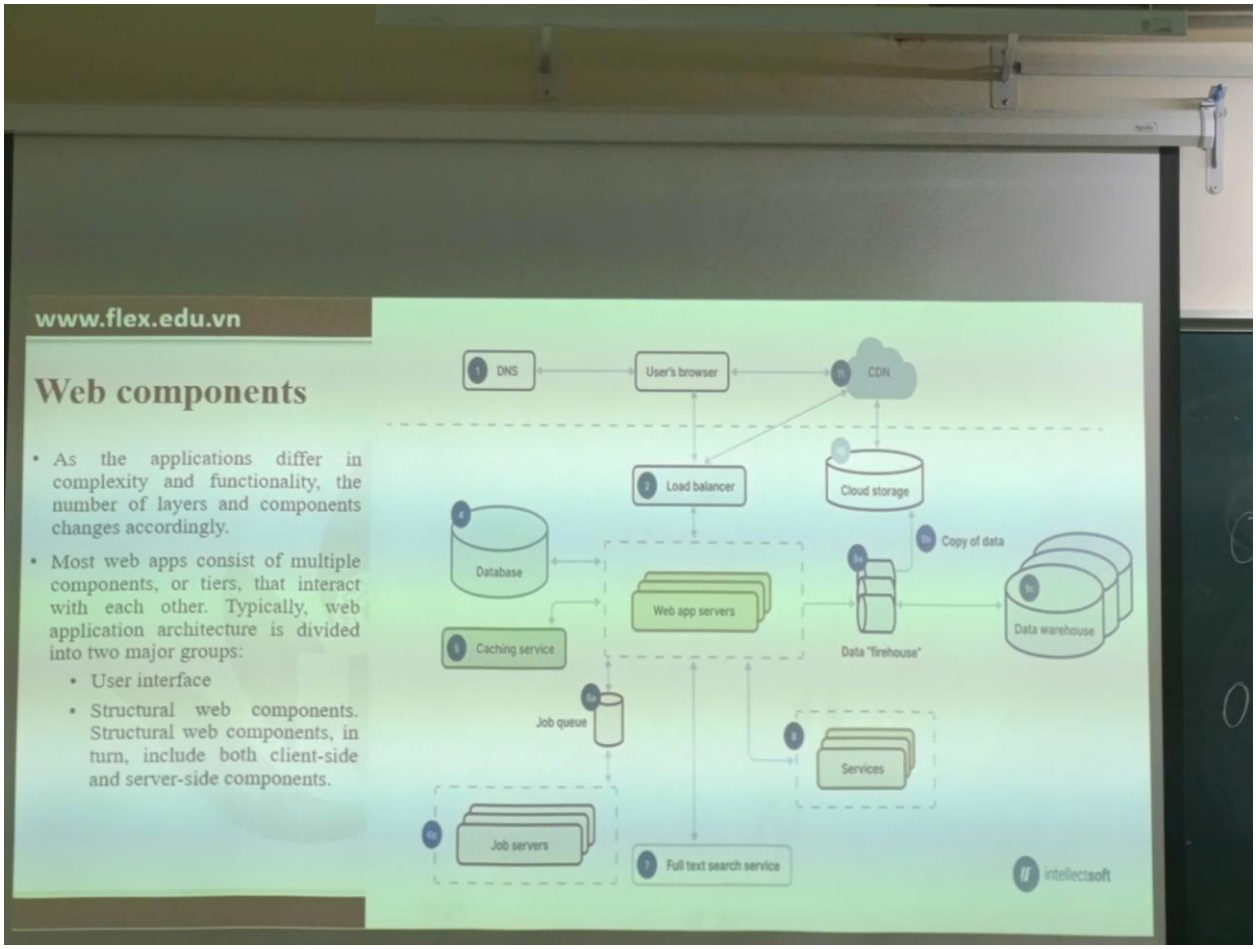

CDN Content Delivery System
Static image, video
Replication
S1
S2
S3
S4
Sapa
Html, CSS, JS
CDN

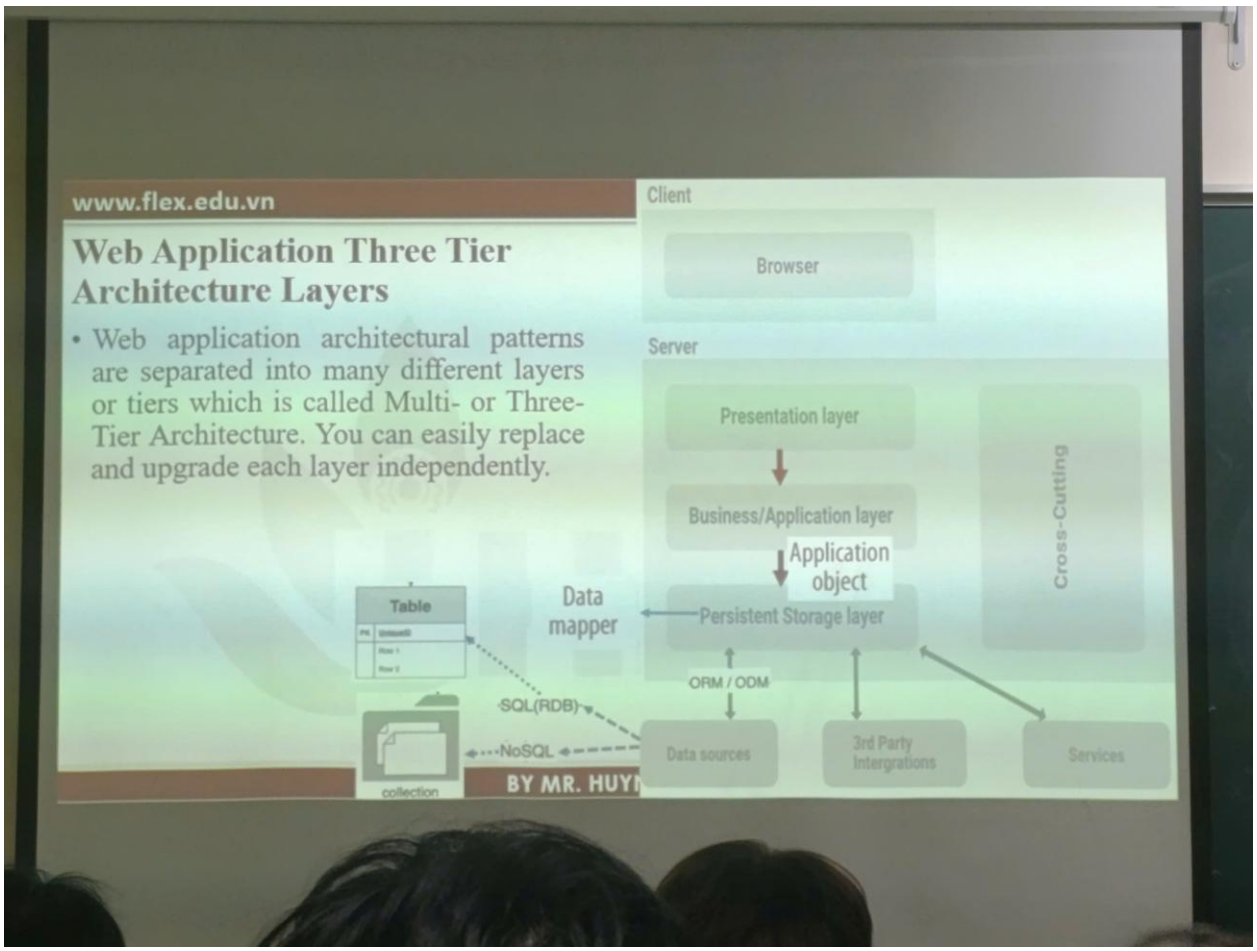

News
WS
overload
en.
de
ELK
Content
elastic
FIFO

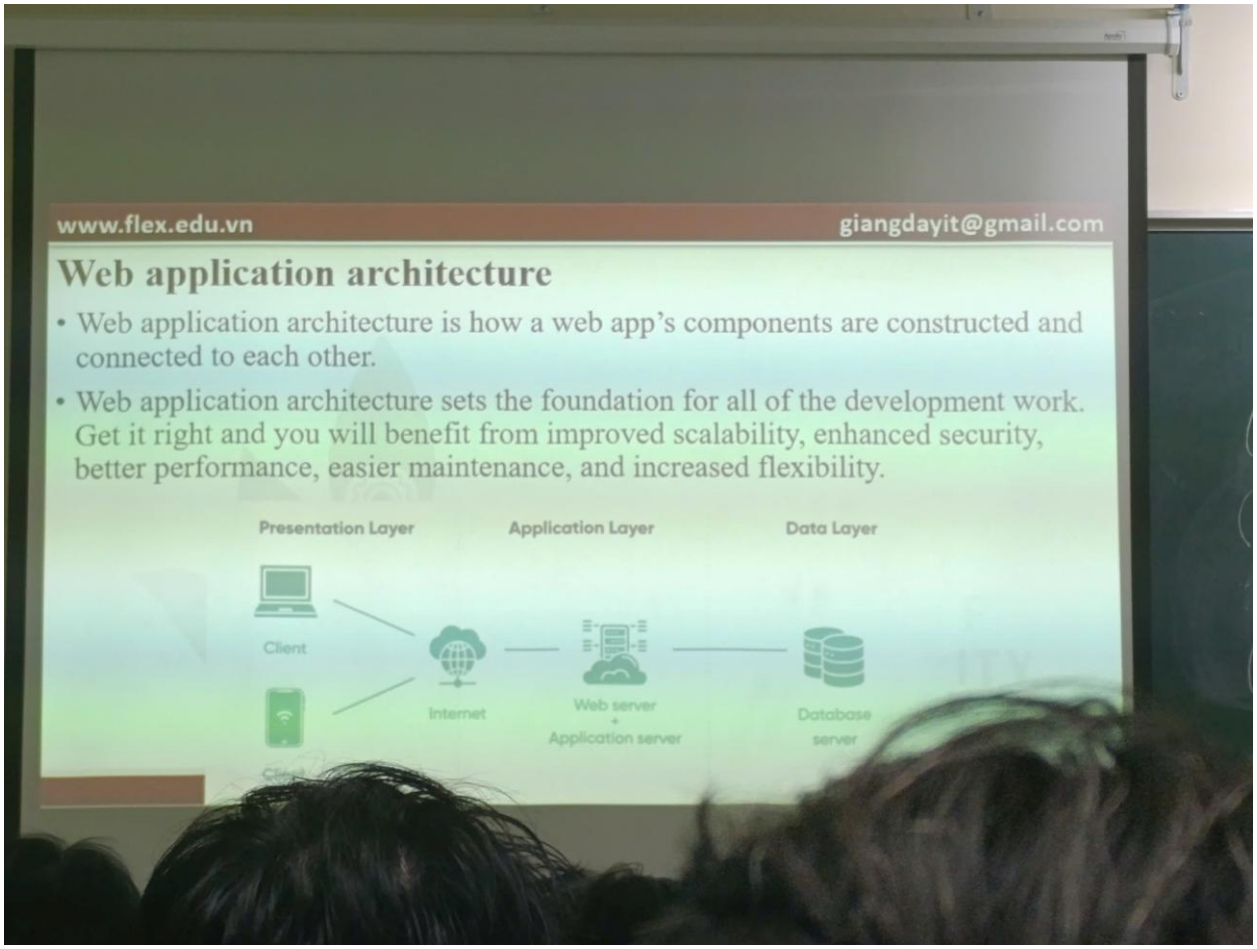

reqs
WS
overload
de
en
FIFO

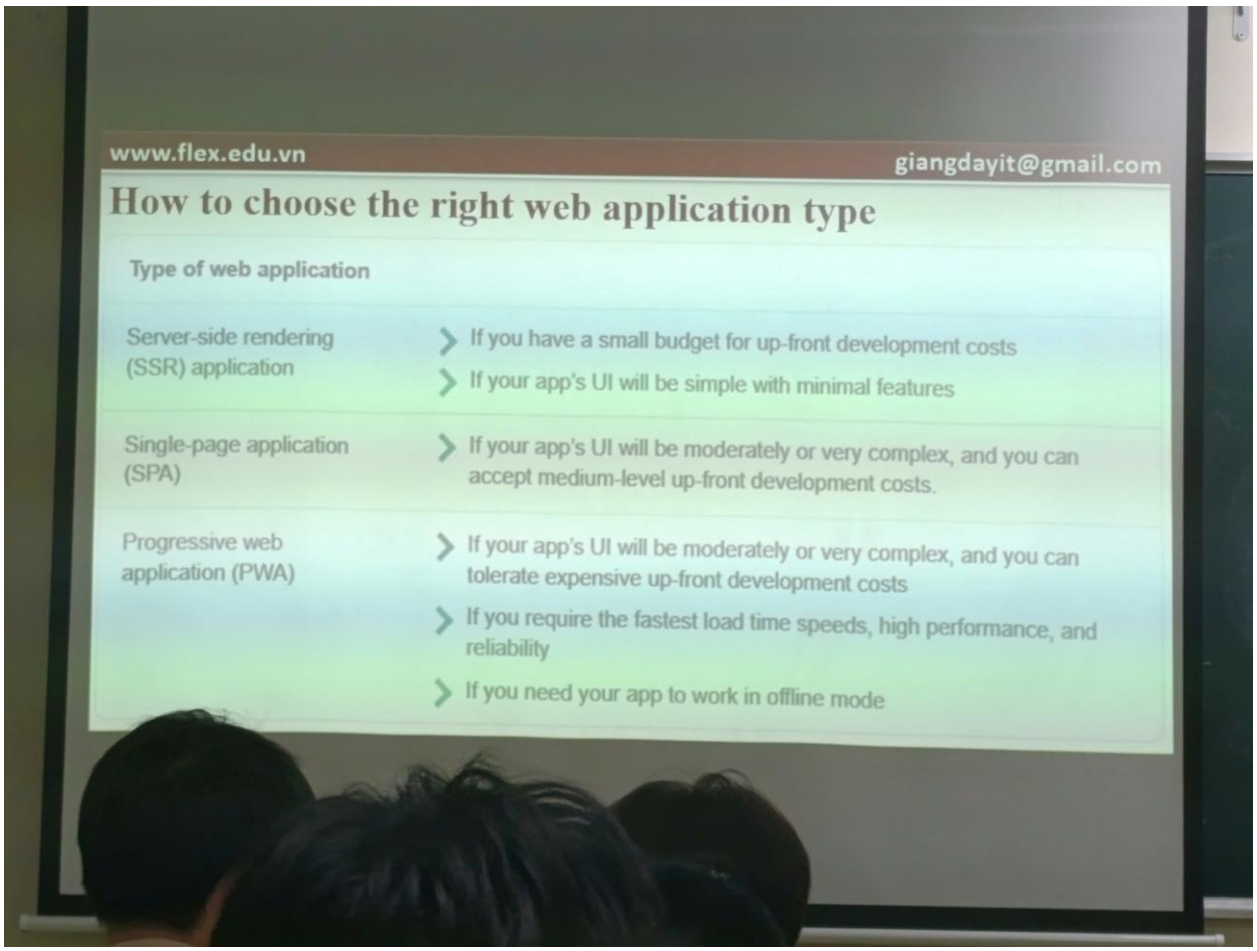

Web Server
num code
Cache
DB

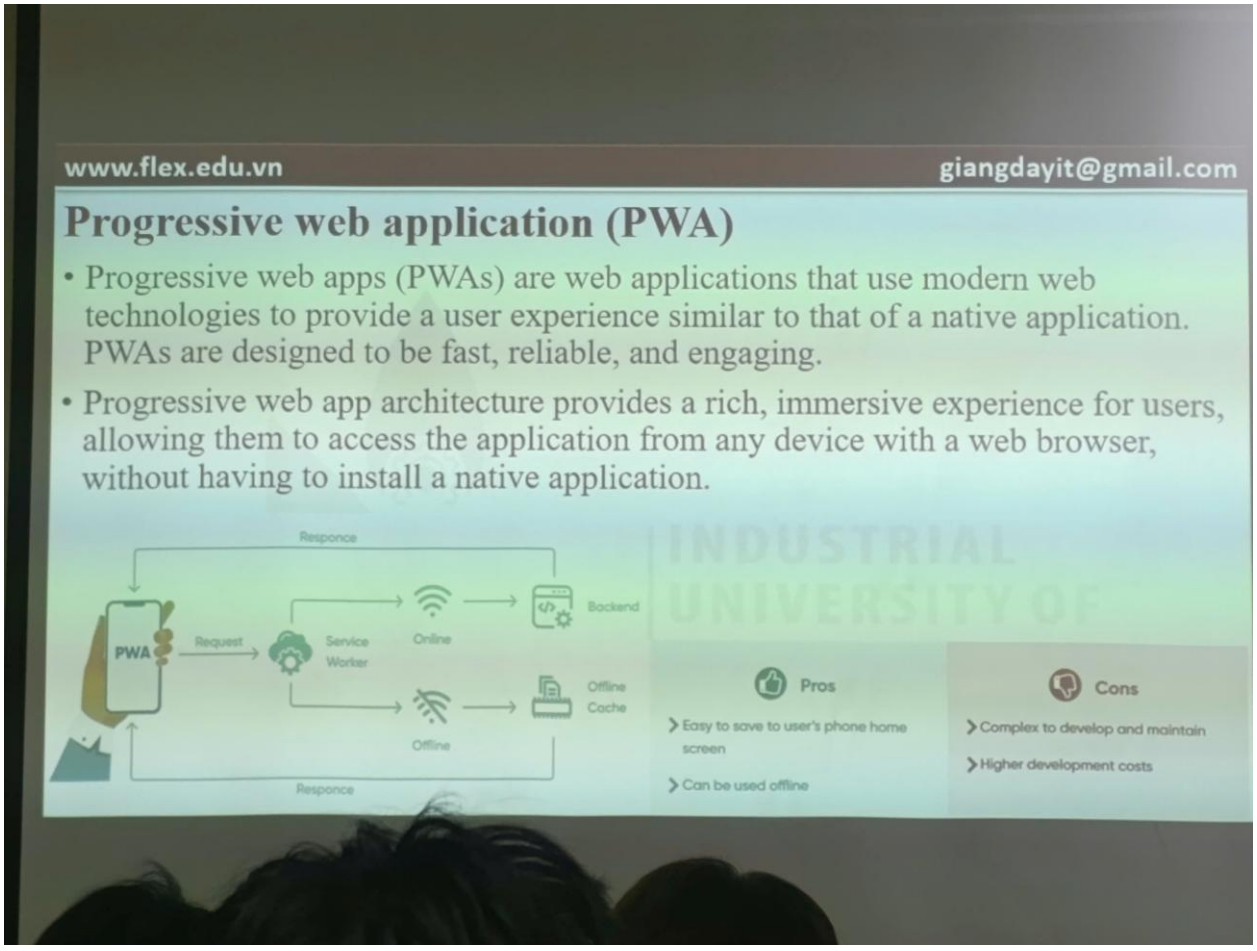

Cross-Cutting
Services
php
Object Relation Mapping
Object Document Mapping
Product { id, name, price }
(Object)
id name price

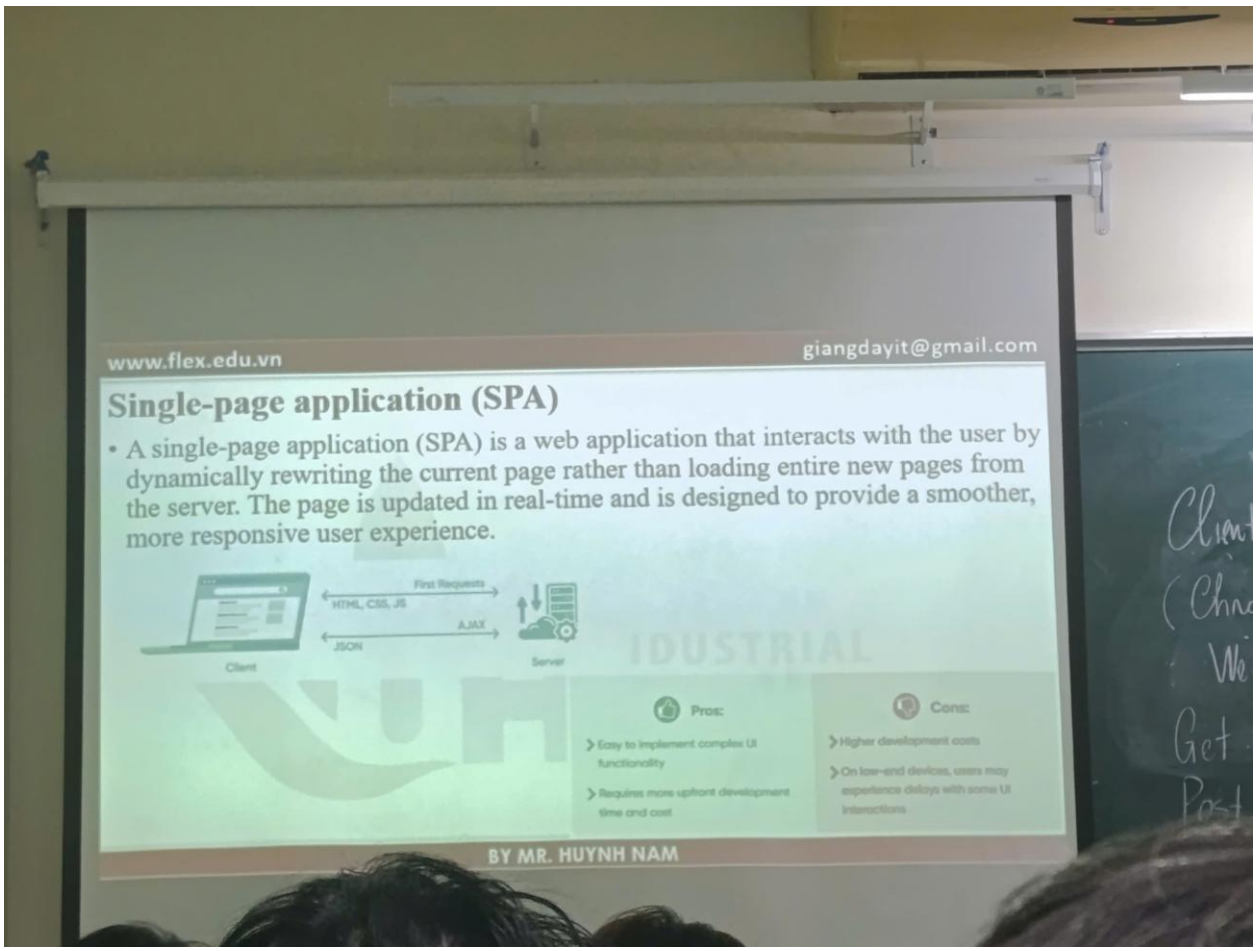

www.flex.edu.vn
Web components
As the applications differ in complexity and functionality, the number of layers and components changes accordingly.
Most web apps consist of multiple components, or tiers, that interact with each other. Typically, web application architecture is divided into two major groups:
User interface
Structural web components. Structural web components, in turn, include both client-side and server-side components.
DNS
User's browser
CDN
Load balancer
Cloud storage
Database
Copy of data
Web app servers
Caching service
Data "firehose"
Data warehouse
Job queue
Services
Job servers
Full text search service
intellectsoft

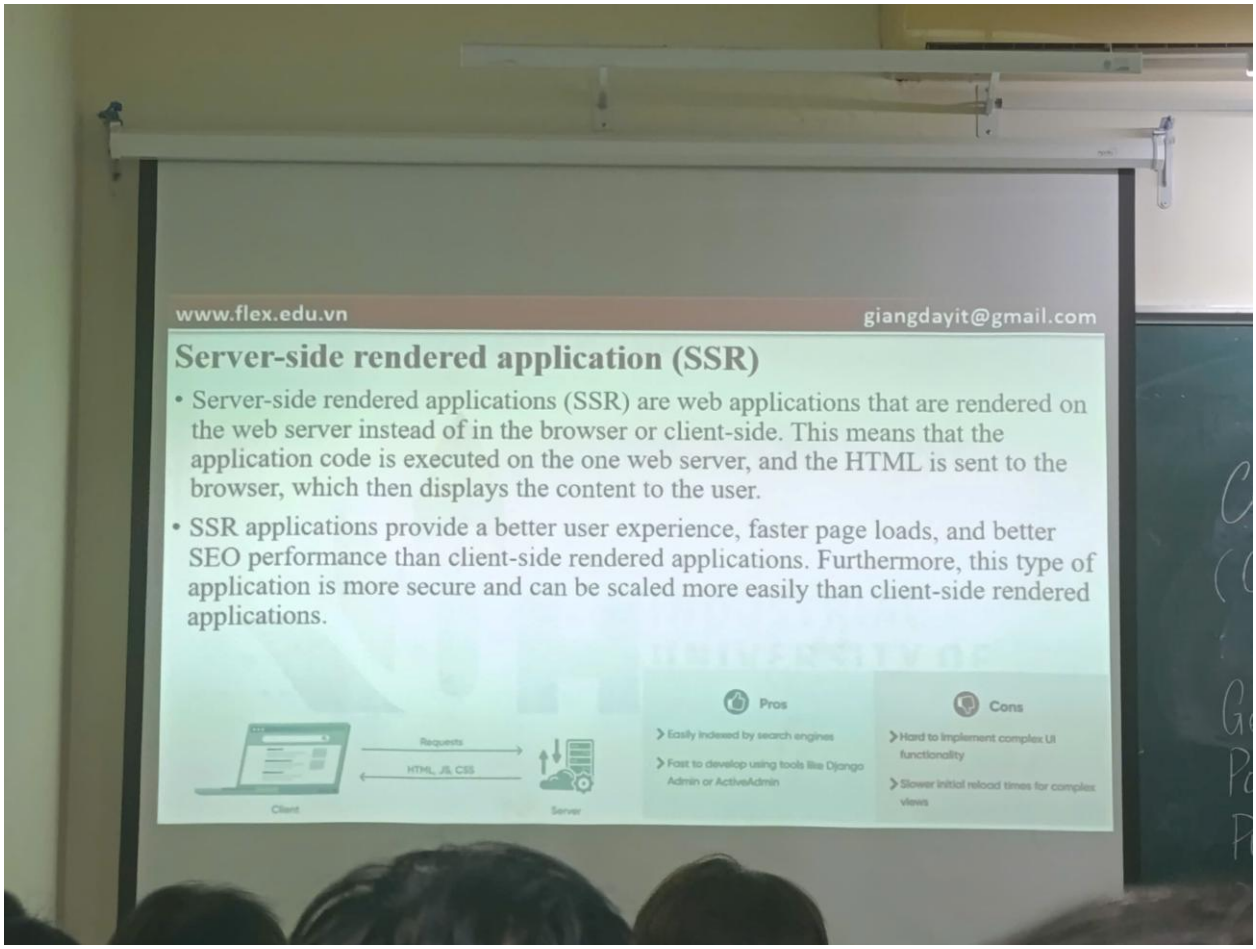

www.flex.edu.vn
Web Application Three Tier Architecture Layers
• Web application architectural patterns are separated into many different layers or tiers which is called Multi- or Three-Tier Architecture. You can easily replace and upgrade each layer independently.
Client
Browser
Server
Presentation layer
Business/Application layer
Application object
Persistent Storage layer
Cross-Cutting
Table
Data mapper
SQL(RDB)
NoSQL
collection
ORM / ODM
Data sources
3rd Party Intergrations
Services
BY MR. HUY

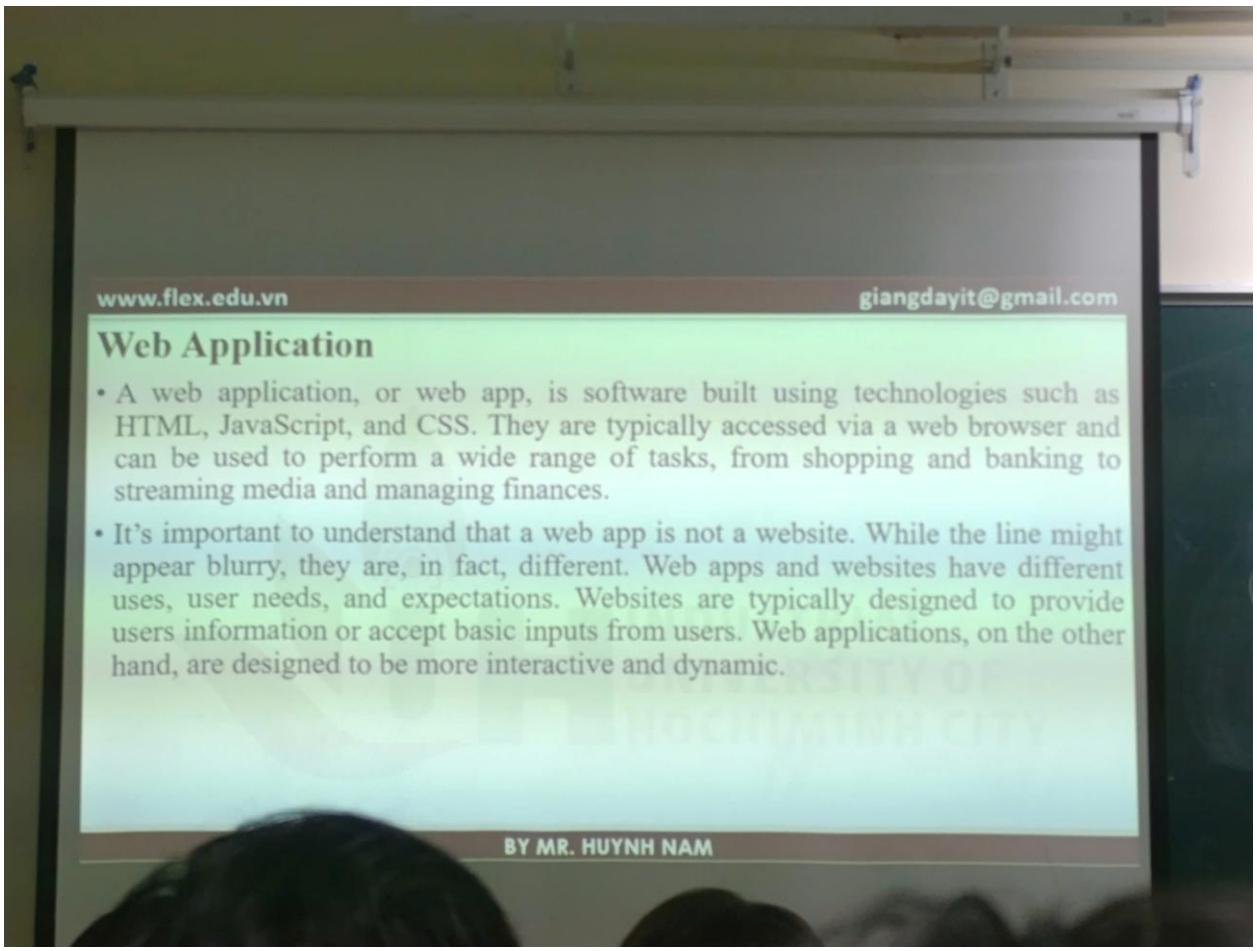

www.flex.edu.vn
giangdayit@gmail.com
Web application architecture
• Web application architecture is how a web app's components are constructed and connected to each other.
• Web application architecture sets the foundation for all of the development work. Get it right and you will benefit from improved scalability, enhanced security, better performance, easier maintenance, and increased flexibility.
Presentation Layer
Application Layer
Data Layer
Client
Internet
Web server
+
Application server
Database server

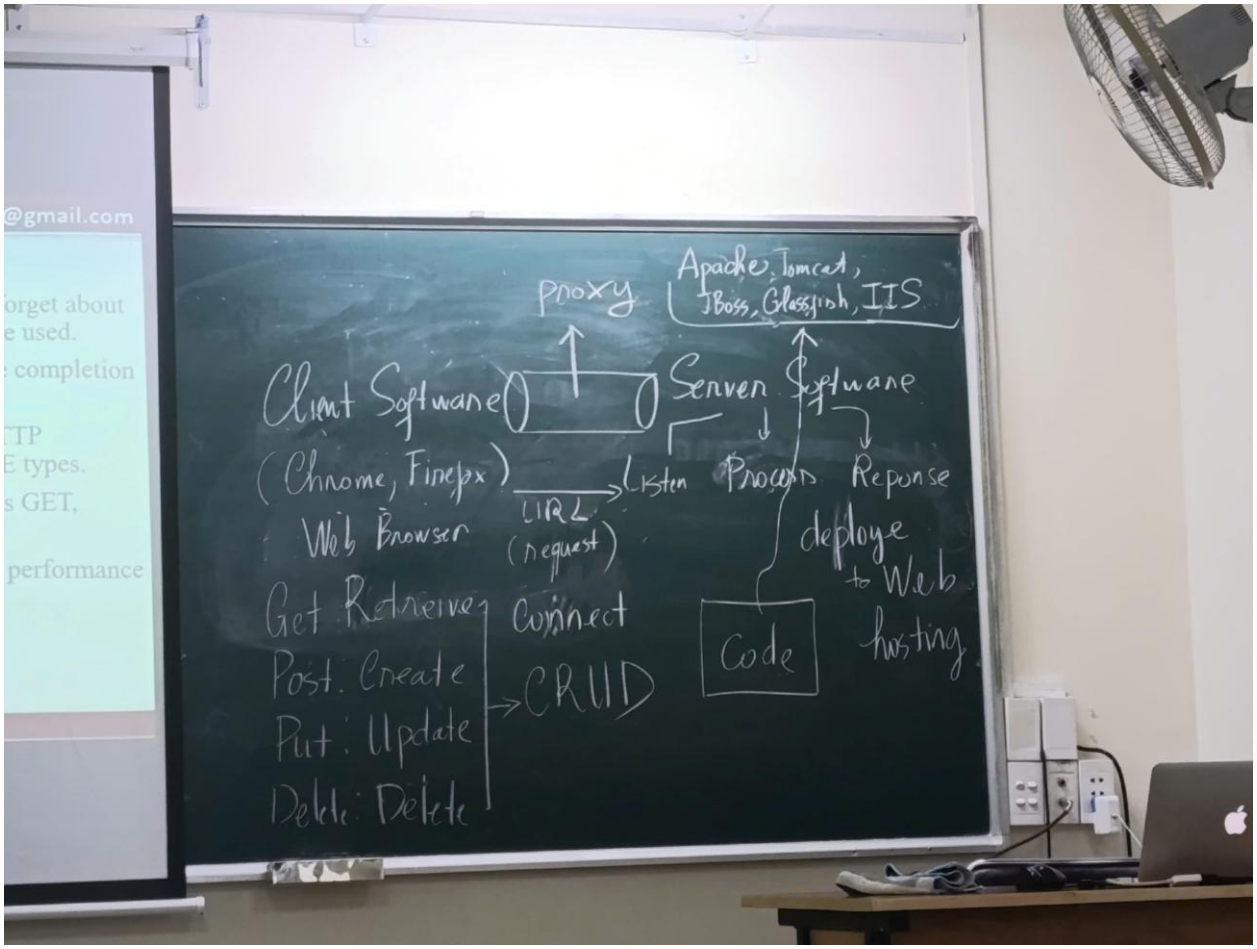

www.flex.edu.vn
giangdayit@gmail.com
How to choose the right web application type
Type of web application
Server-side rendering (SSR) application
If you have a small budget for up-front development costs
If your app's UI will be simple with minimal features
Single-page application (SPA)
If your app's UI will be moderately or very complex, and you can accept medium-level up-front development costs.
Progressive web application (PWA)
If your app's UI will be moderately or very complex, and you can tolerate expensive up-front development costs
If you require the fastest load time speeds, high performance, and reliability
If you need your app to work in offline mode

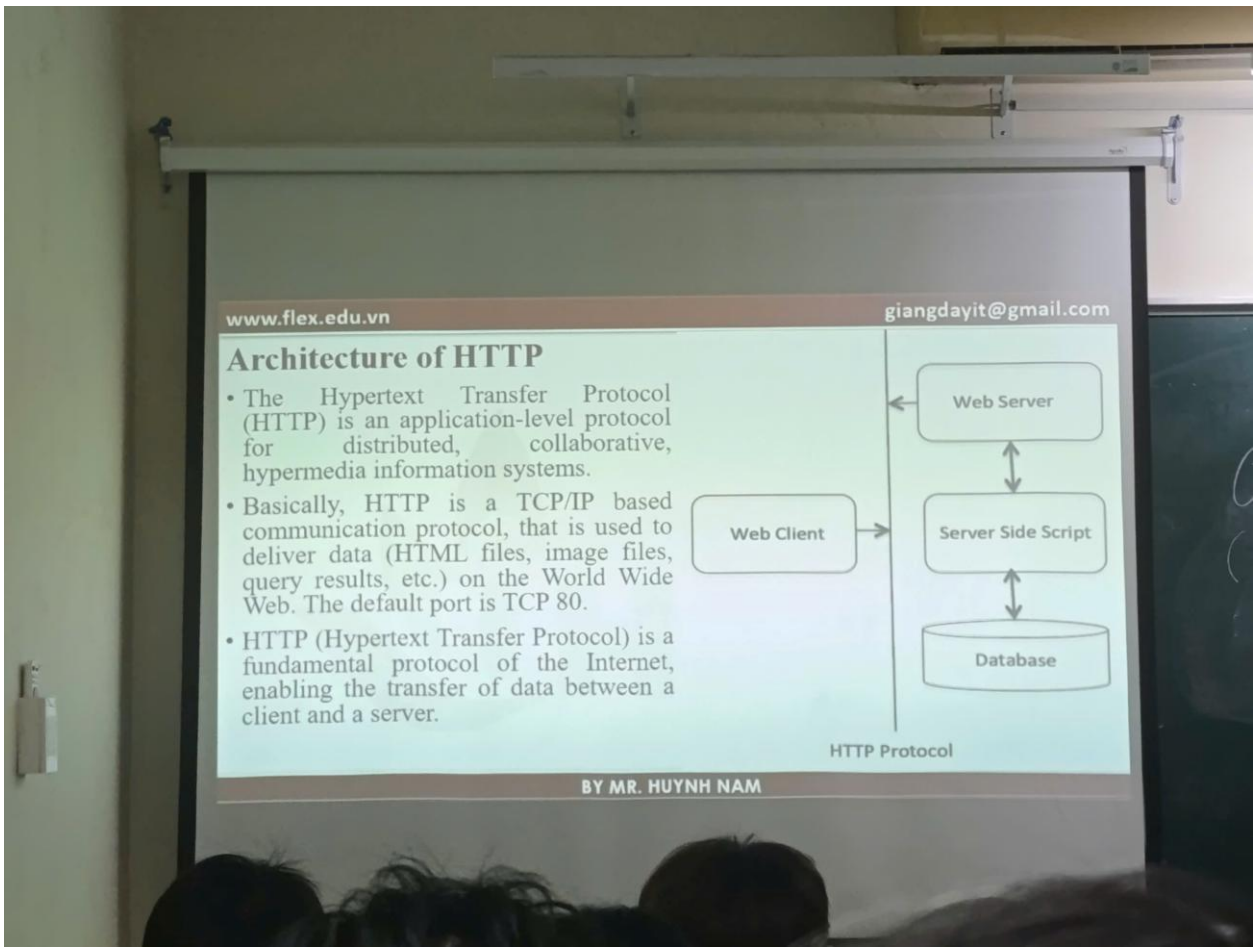

www.flex.edu.vn
giangdayit@gmail.com
Progressive web application (PWA)
• Progressive web apps (PWAs) are web applications that use modern web technologies to provide a user experience similar to that of a native application. PWAs are designed to be fast, reliable, and engaging.
• Progressive web app architecture provides a rich, immersive experience for users, allowing them to access the application from any device with a web browser, without having to install a native application.
Responce
PWA
Request
Service Worker
Online
Offline
Backend
Offline Cache
Responce
Pros
Easy to save to user's phone home screen
Can be used offline
Cons
Complex to develop and maintain
Higher development costs

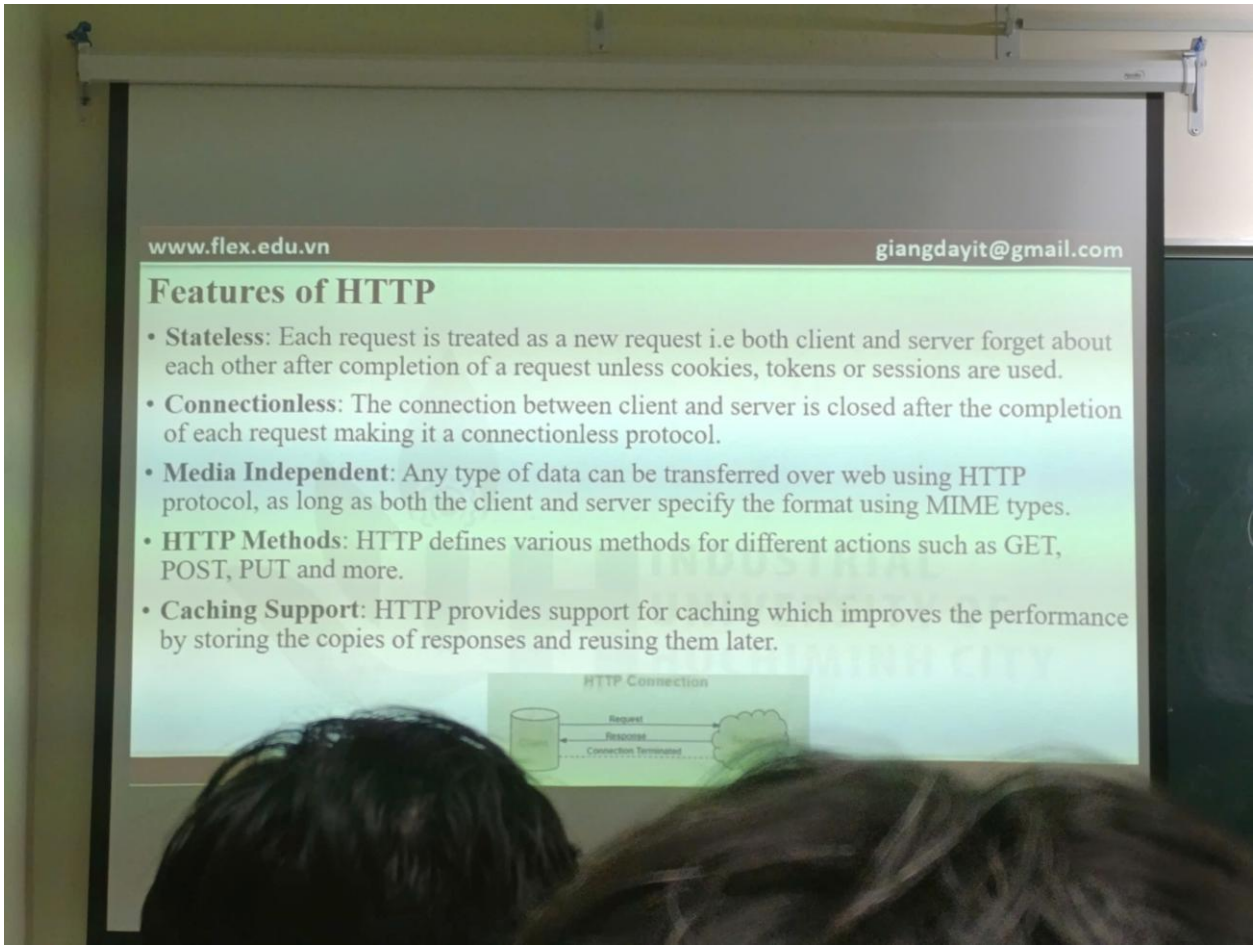

www.flex.edu.vn
giangdayit@gmail.com
Single-page application (SPA)
• A single-page application (SPA) is a web application that interacts with the user by dynamically rewriting the current page rather than loading entire new pages from the server. The page is updated in real-time and is designed to provide a smoother, more responsive user experience.
First Requests
HTML, CSS, JS
AJAX
JSON
Client
Server
Pros:
Easy to implement complex UI functionality
Requires more upfront development time and cost
Cons:
Higher development costs
On low-end devices, users may experience delays with some UI interactions
BY MR. HUYNH NAM


www.flex.edu.vn
giangdayit@gmail.com
Server-side rendered application (SSR)
• Server-side rendered applications (SSR) are web applications that are rendered on the web server instead of in the browser or client-side. This means that the application code is executed on the one web server, and the HTML is sent to the browser, which then displays the content to the user.
• SSR applications provide a better user experience, faster page loads, and better SEO performance than client-side rendered applications. Furthermore, this type of application is more secure and can be scaled more easily than client-side rendered applications.
Requests
HTML, JS, CSS
Client
Server
Pros
Easily indexed by search engines
Fast to develop using tools like Django Admin or ActiveAdmin
Cons
Hard to implement complex UI functionality
Slower initial reload times for complex views


www.flex.edu.vn
giangdayit@gmail.com
Web Application
• A web application, or web app, is software built using technologies such as HTML, JavaScript, and CSS. They are typically accessed via a web browser and can be used to perform a wide range of tasks, from shopping and banking to streaming media and managing finances.
• It's important to understand that a web app is not a website. While the line might appear blurry, they are, in fact, different. Web apps and websites have different uses, user needs, and expectations. Websites are typically designed to provide users information or accept basic inputs from users. Web applications, on the other hand, are designed to be more interactive and dynamic.
BY MR. HUYNH NAM


Proxy
Apache, Tomcat, JBoss, Glassfish, IIS
Client Software
Server Software
(Chrome, Firefox)
Web Browser
URL (request)
Listen
Process
Reponse
deploye to Web hosting
Get: Retrieve
Post: Create
Put: Update
Delete: Delete
Connect
CRUD
Code


www.flex.edu.vn
giangdayit@gmail.com
Architecture of HTTP
• The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems.
• Basically, HTTP is a TCP/IP based communication protocol, that is used to deliver data (HTML files, image files, query results, etc.) on the World Wide Web. The default port is TCP 80.
• HTTP (Hypertext Transfer Protocol) is a fundamental protocol of the Internet, enabling the transfer of data between a client and a server.
Web Client
Web Server
Server Side Script
Database
HTTP Protocol
BY MR. HUYNH NAM


www.flex.edu.vn
giangdayit@gmail.com
Features of HTTP
• Stateless: Each request is treated as a new request i.e both client and server forget about each other after completion of a request unless cookies, tokens or sessions are used.
• Connectionless: The connection between client and server is closed after the completion of each request making it a connectionless protocol.
• Media Independent: Any type of data can be transferred over web using HTTP protocol, as long as both the client and server specify the format using MIME types.
• HTTP Methods: HTTP defines various methods for different actions such as GET, POST, PUT and more.
• Caching Support: HTTP provides support for caching which improves the performance by storing the copies of responses and reusing them later.
HTTP Connection