UNIT 1 BASICS OF FULL STACK

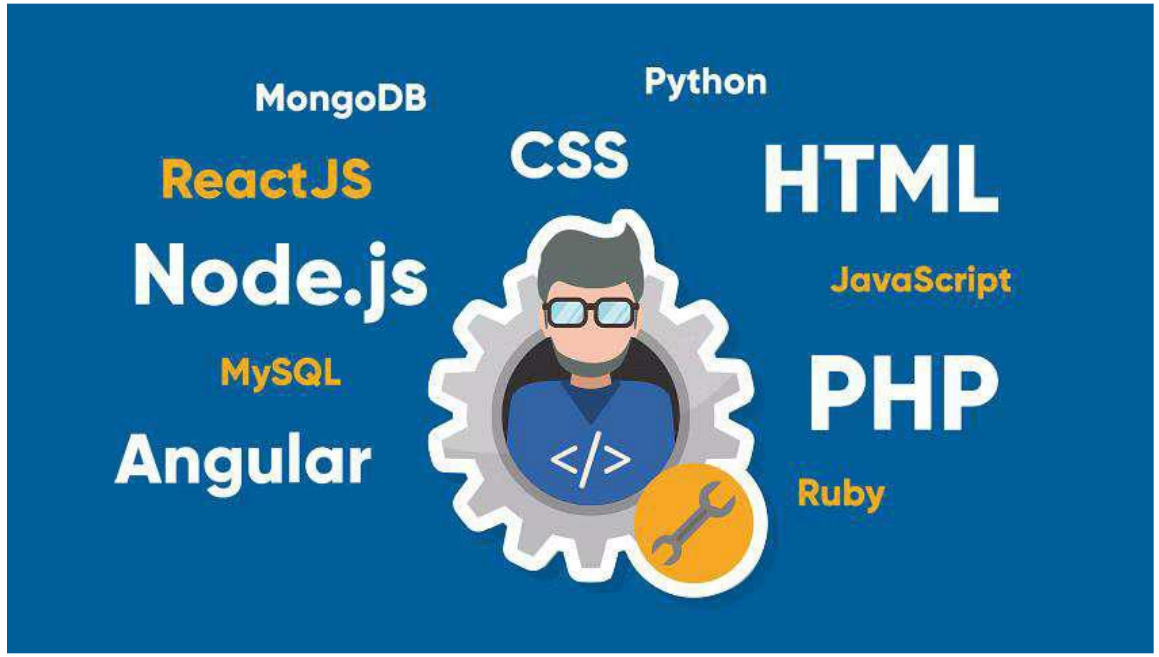
Understanding the Basic Web Development Framework - User - Browser – Webserver – Backend Services – MVC Architecture - Understanding the different stacks –The role of Express – Angular – Node – Mongo DB – React

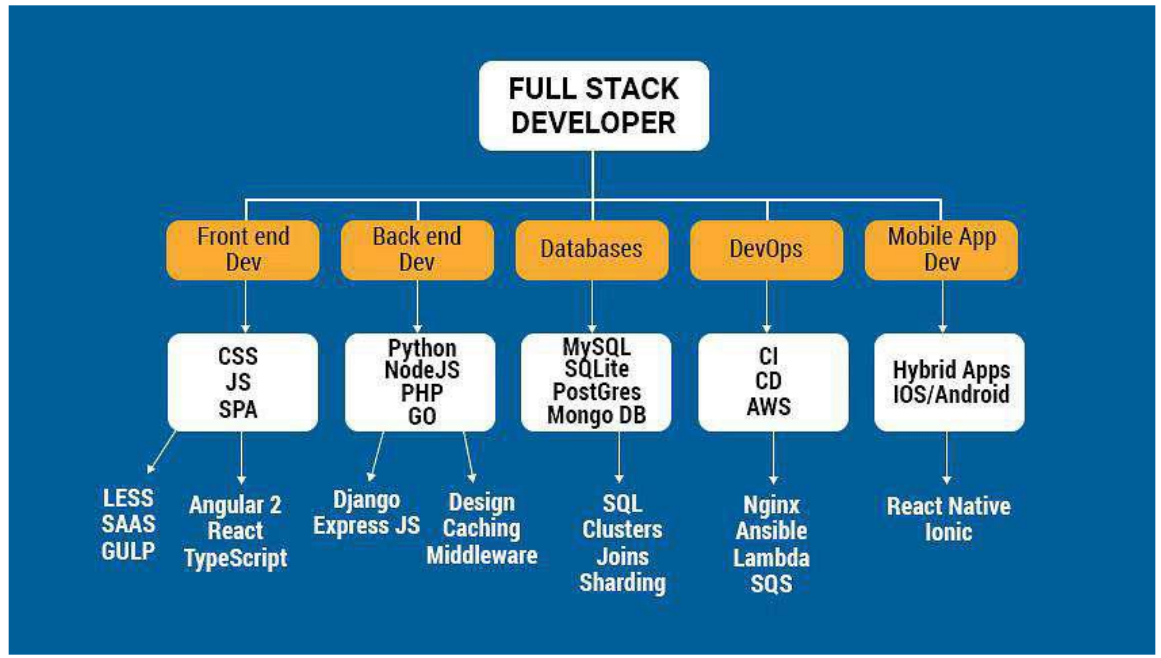
Full Stack Web Developer

• A full stack web developer is a person who can develop both **client** and **server** software.

In addition to mastering HTML and CSS, he/she also knows how to:
• Program a **browser** (like using JavaScript, jQuery, Angular, or Vue)
• Program a **server** (like using PHP, ASP, Python, or Node)
• Program a **database** (like using SQL, SQLite, or MongoDB)

Skills required for a full stack developer




FULL STACK DEVELOPER
Front end Dev
Back end Dev
Databases
DevOps
Mobile App Dev
CSS
JS
SPA
Python
NodeJS
PHP
GO
MySQL
SQLite
PostGres
Mongo DB
CI
CD
AWS
Hybrid Apps
IOS/Android
LESS
SAAS
GULP
Angular 2
React
TypeScript
Django
Express JS
Design
Caching
Middleware
SQL
Clusters
Joins
Sharding
Nginx
Ansible
Lambda
SQS
React Native
Ionic

Popular stacks

MEAN stack includes the following:

• Front-end framework: JavaScript and AngularJS
• Database: MongoDB
•Web-framework: Node.js
• Back-end web framework: Express.js

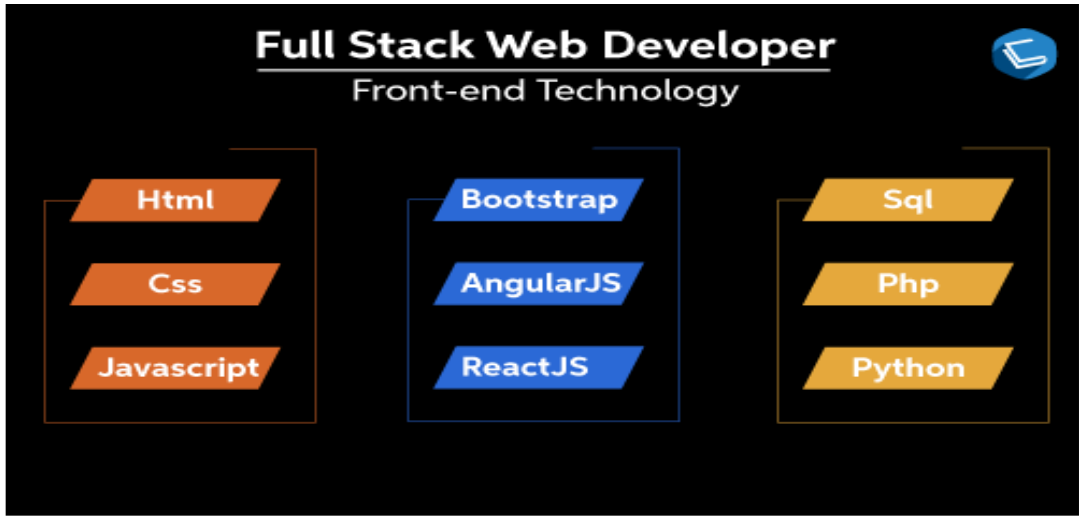
The **LAMP stack** is made up of

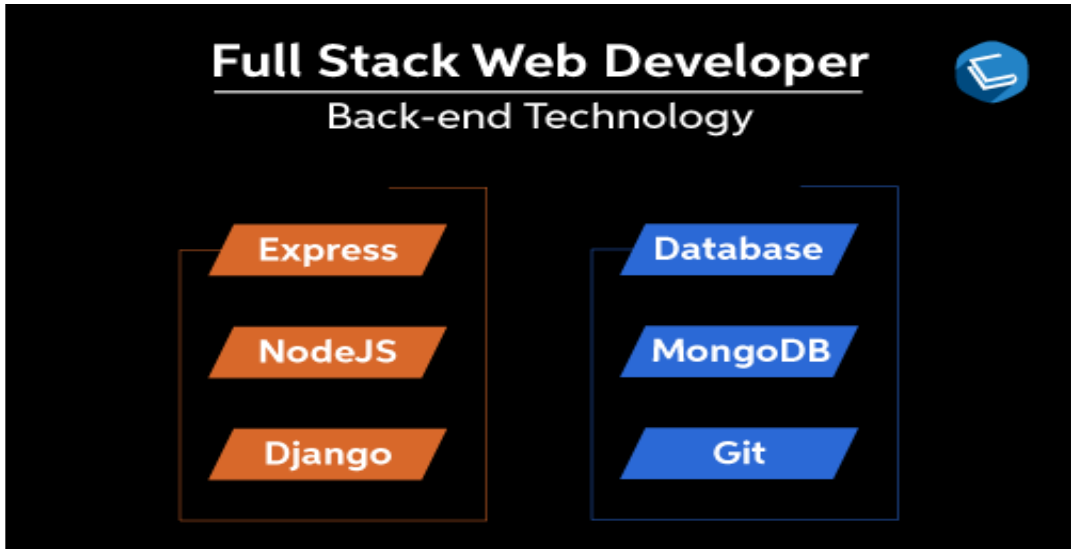
- Operating System: Linux
- Web Server: Apache
- Front-end framework: JavaScript
- Database: MySQL
- Programming and Development: PHP

MERN stack, like MEAN stack, consists of JavaScript based technologies.

- These are the main components of MERN stack:
- Database: MongoDB
- Web Programming Framework: Express
- Building UI: Node.js and React

Full stack web development road map

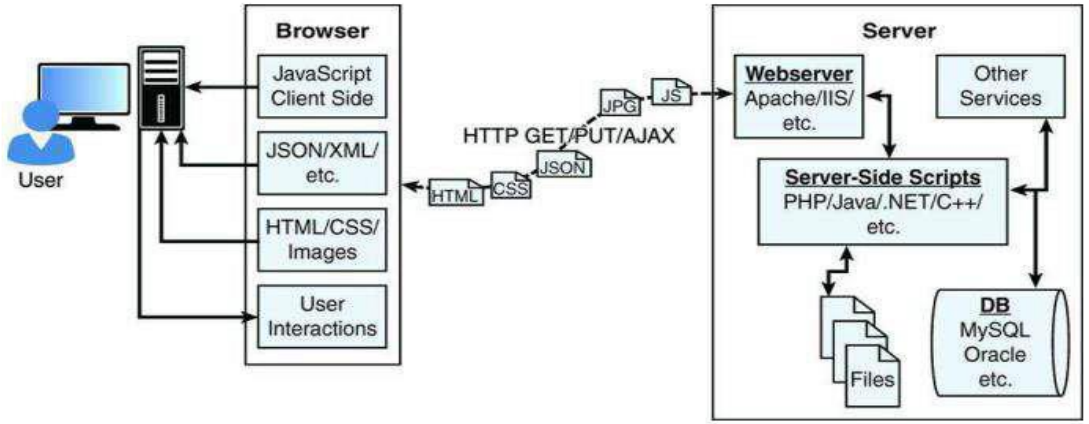


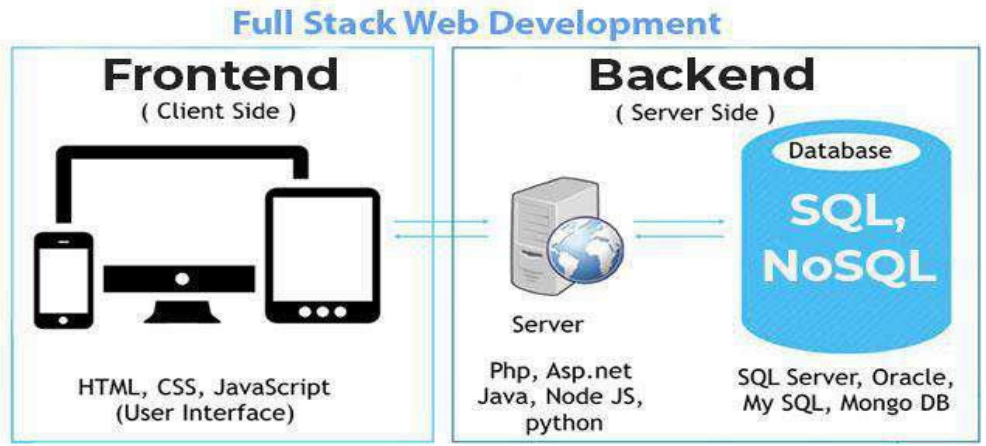

Full Stack Web Developer
Back-end Technology
Express
NodeJS
Django
Database
MongoDB
Git

Understanding the Basic Web Development Framework

- A web development framework is a set of resources and tools for software developers to build and manage web applications, web services and websites, as well as to develop application programming interfaces (APIs).
- Web development frameworks are also referred to as web application frameworks or simply web frameworks.

The main components of any given web framework are the user, browser, webserver, and backend services.




Full Stack Web Development
Frontend
(Client Side)
HTML, CSS, JavaScript
(User Interface)
Backend
(Server Side)
Server
Php, Asp.net
Java, Node JS,
python
Database
SQL,
NoSQL
SQL Server, Oracle,
My SQL, Mongo DB

User

- fundamental part of all websites
- User expectations – developing a good website.
- Users accept “world-widewait,” but no longer.
- They expect websites to behave closer to computers and mobile applications.
- role - to sit on the visual output and interaction input of webpages.

Browser

- The browser plays three roles in the web framework.
- First, it provides communication to and from the webserver.
- Second, it interprets the data from the server and renders it into the view that the user actually sees.
- Finally, the browser handles user interaction through the keyboard, mouse, touchscreen, or other input device and takes the appropriate action.

Browser to Webserver Communication

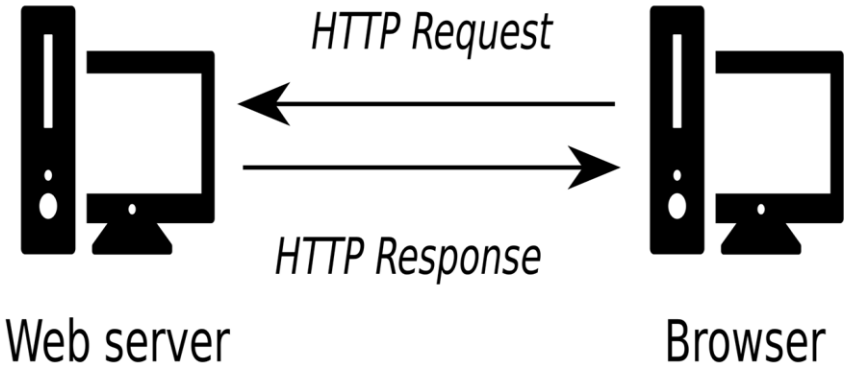
Browser-to-webserver communication consists of a series of requests using the HTTP and HTTPS protocols.

The browser makes three main types of requests to the server:

GET: The GET request is typically used to retrieve data from the server, such as .html files, images, or JSON data.

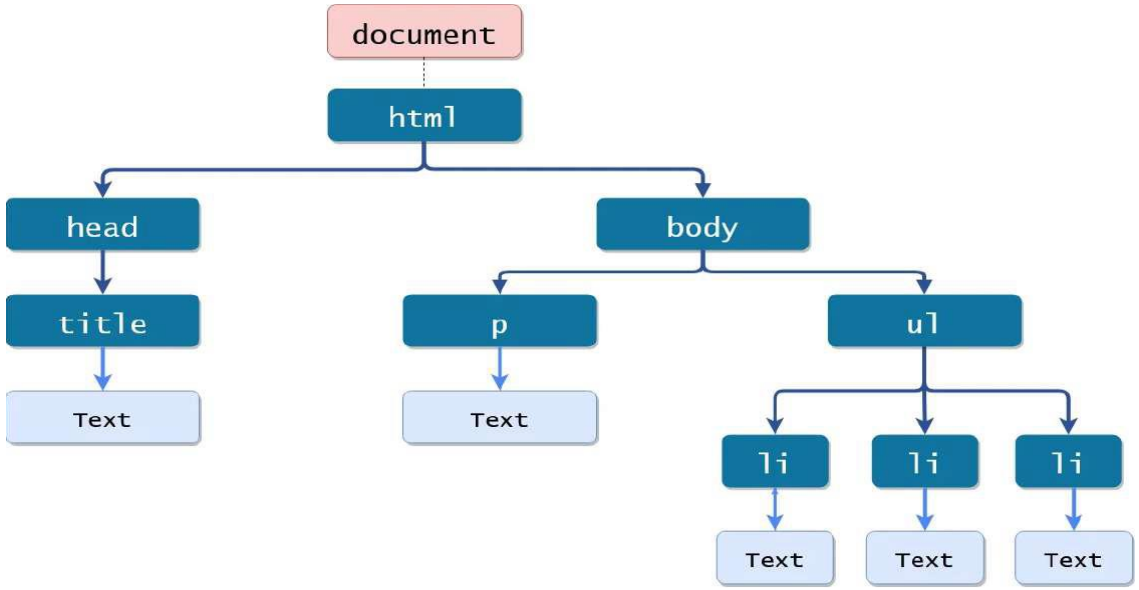
POST: POST requests are used when sending data to
the server, such as adding an item to a shopping cart or submitting a web form.

AJAX: Asynchronous JavaScript and XML (AJAX) is just a GET or POST request done directly by JavaScript running in the browser.



Rendering the Browser View

- The browser reads data from the initial URL and then renders the HTML document to build a Document Object Model (DOM).
- The DOM is a tree structure object with the HTML document as the root. The structure of the tree basically matches the structure of the HTML document.
- For example, the document will have html as a child, and html will have head and body as children, and body may have div, p, or other elements as children, like this:
- document
- + html
- + head
- + body
- + div
- + p



The following are the most common types of data the browser uses.

- **HTML files:** These provide the fundamental structure of the DOM.
- **CSS files:** These define how each of the elements on the page is to be styled; for example, font, color, borders, and spacing.
- **Client-side scripts**: These are typically JavaScript files. They can provide added functionality, manipulate the DOM to change the look of the webpage, and provide any necessary logic required to display the page.
- **Media files:** Image, video, and sound files are rendered as part of the webpage.
- **Data:** Any data, such as XML, JSON, or raw text, can be provided by the webserver as a response to an AJAX request.
- **HTTP headers:** The HTTP protocol defines a set of headers that can be used by the browser and client-side scripts to define the behavior of the webpage

Webserver

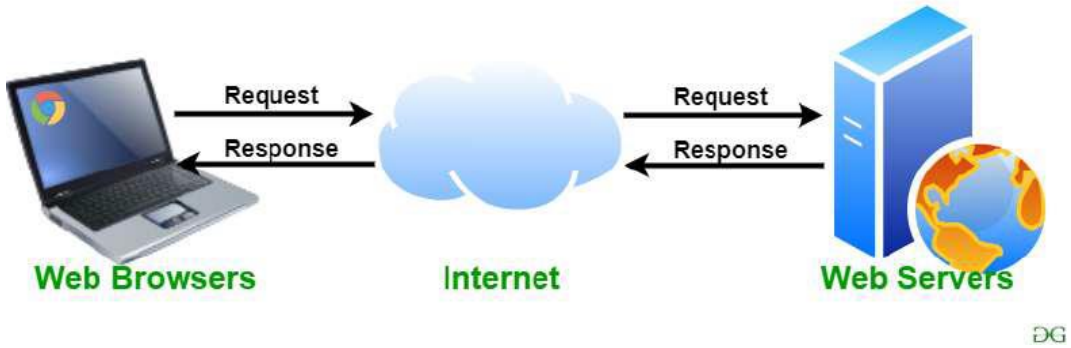
• The webserver's main focus is handling requests from browsers.

• Most out-of-the-box webservers, such as Apache and IIS, are made to serve static files such as .html, .css, and media files.

• A *server-side program* can be executed by the webserver to perform the task the browser is requesting.

• These can be written in PHP, Python, C, C++, C#, Java, … the list goes on and on.

• Webservers such as Apache and IIS provide mechanisms to include server-side scripts.



Backend Services

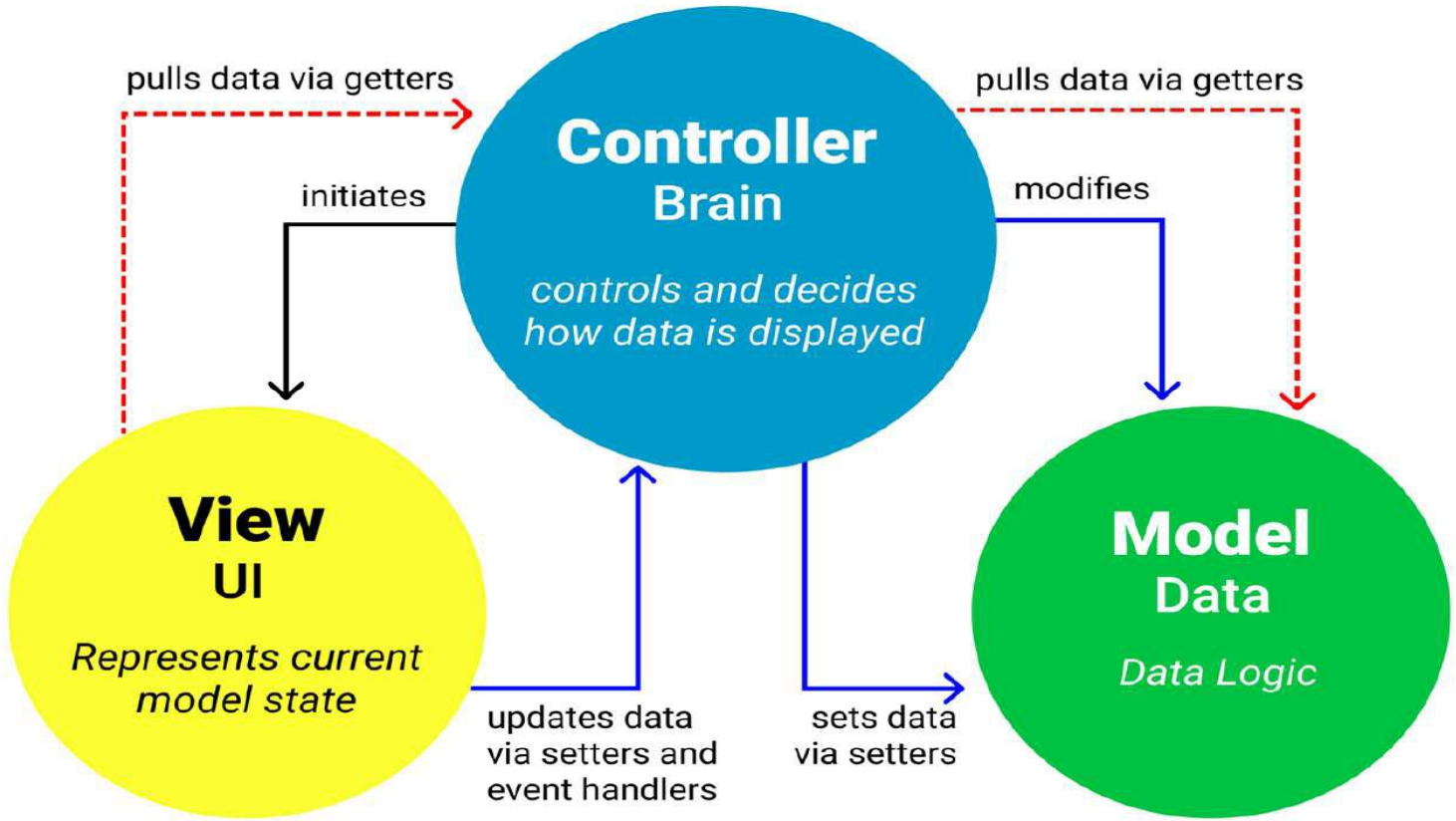
• Run behind the webserver and provide data used to build responses to the browser.

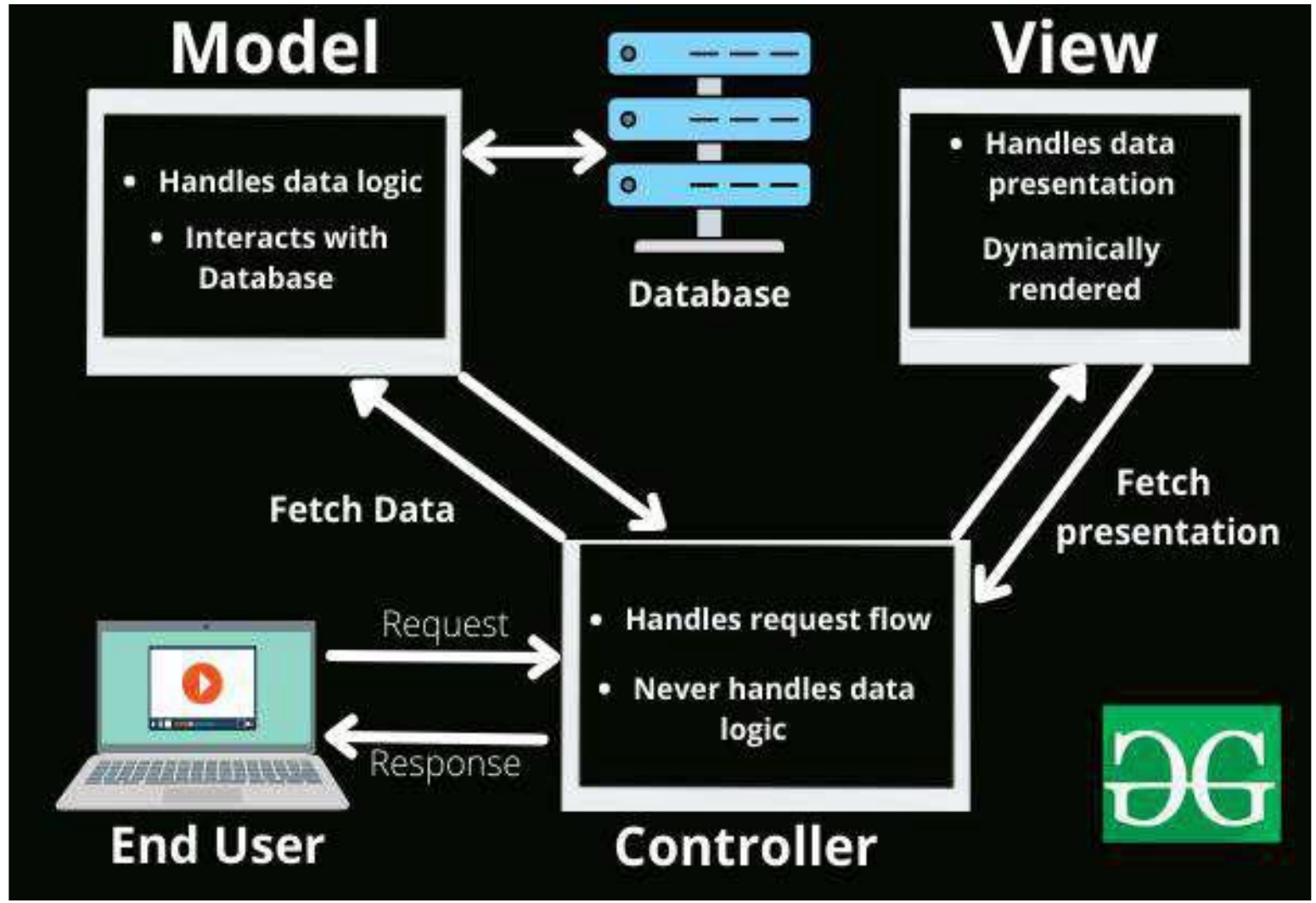
• Common type of backend service is a database that stores information.

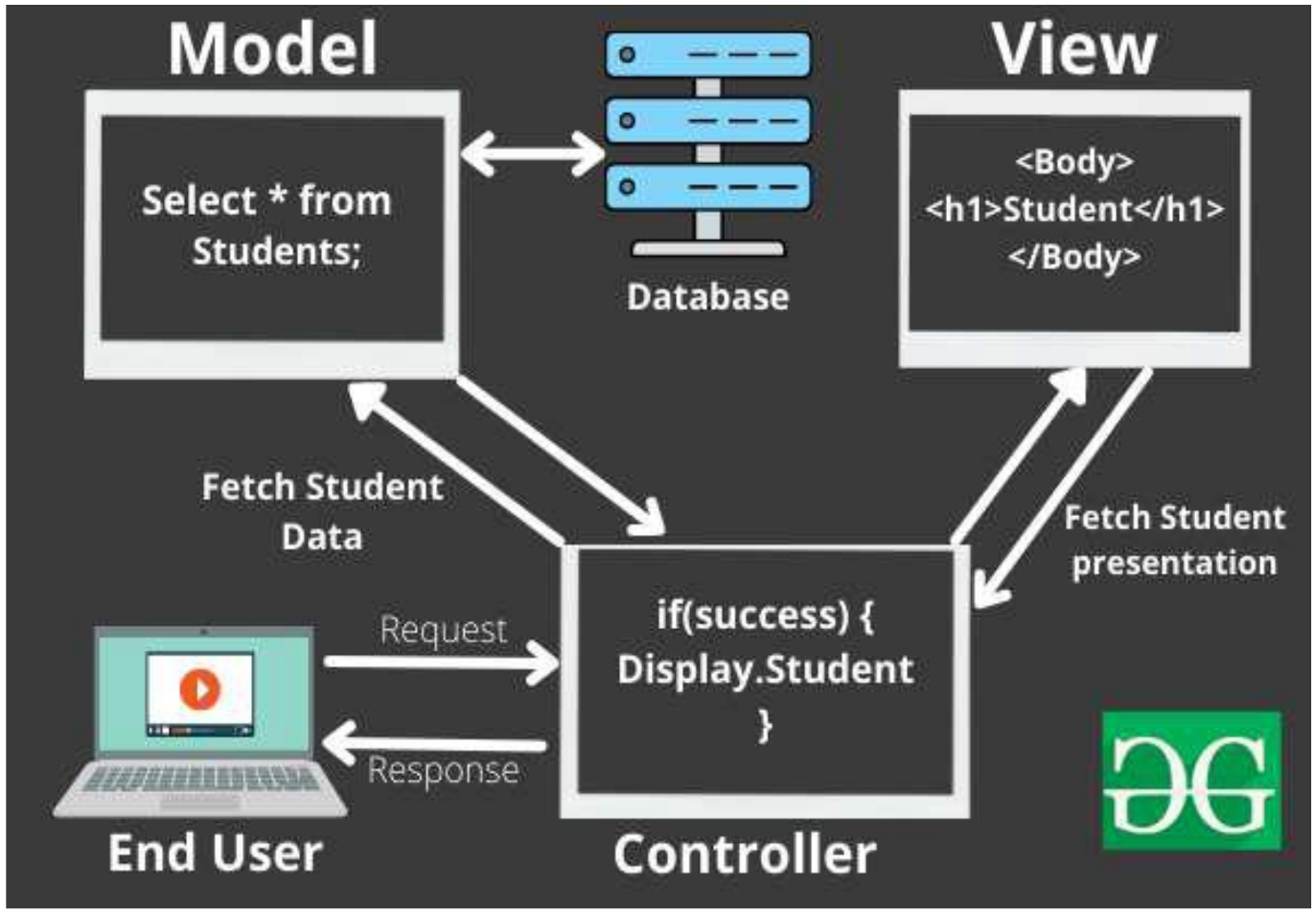
• When a request comes in from the browser that requires information from the database or other backend service, the server-side script connects to the database, retrieves the information, formats it, and then sends it back to the browser.

• Conversely, when data comes in from a web request that needs to be stored in the database, the server-side script connects to the database and updates the data.

MVC Architecture Pattern

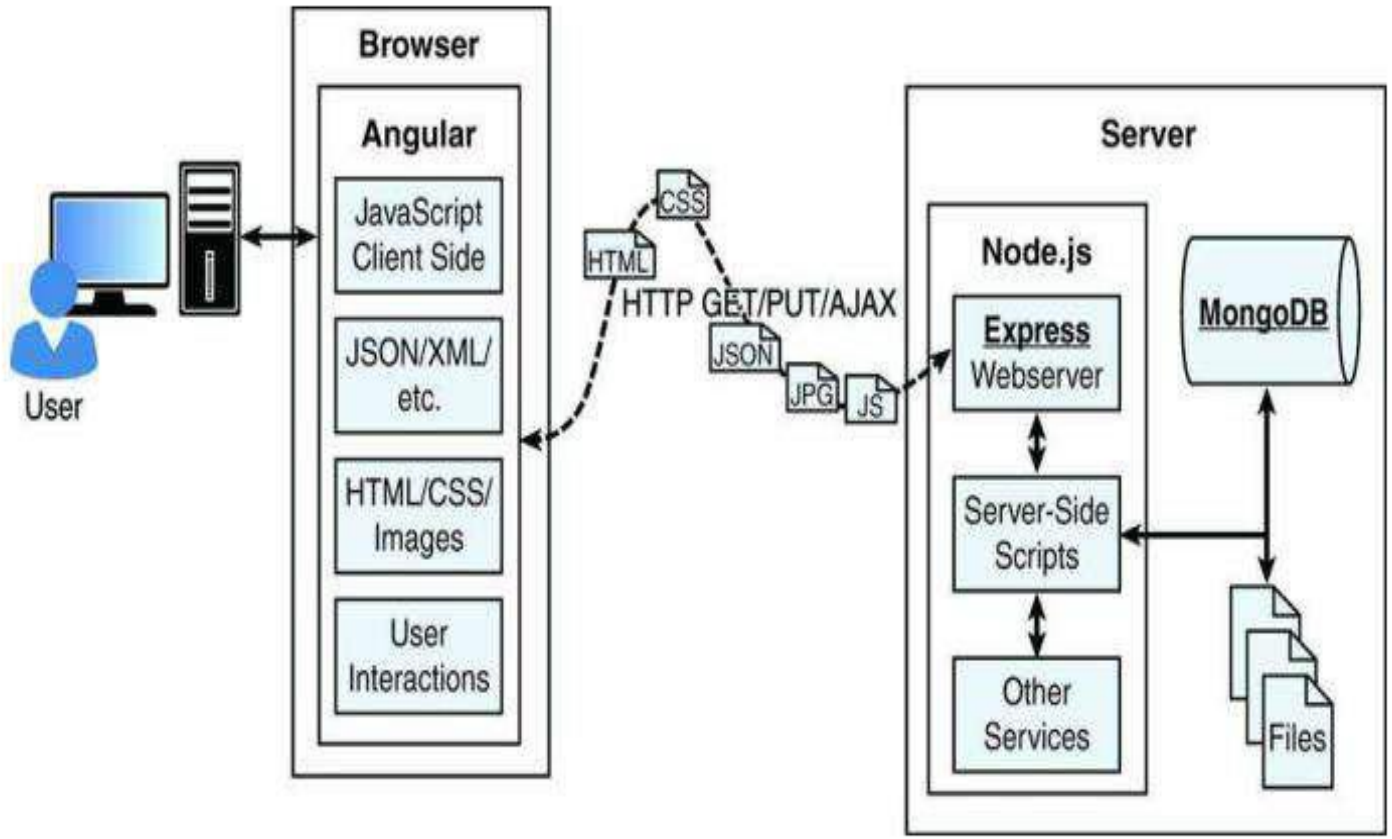



Model
Handles data logic
Interacts with Database
Database
View
Handles data presentation
Dynamically rendered
Fetch Data
Fetch presentation
Request
Response
Handles request flow
Never handles data logic
End User
Controller


Model
Select * from Students;
Database
View
<Body>
<h1>Student</h1>
</Body>
Fetch Student Data
Fetch Student presentation
Request
Response
if(success) {
Display.Student
}
End User
Controller

UNDERSTANDING THE DIFFERENT STACKS

• The most common version of the stack is the **Node.js-to-Angular stack** comprised of **MongoDB, Express, Angular, and Node.js**.
• In the **Node.js-to-Angular stack**,
• Node.js provides the fundamental platform for development.
• The backend services and server-side scripts are all written in Node.js.
• MongoDB provides the data store for the website
• The webserver is defined by Express
• The view in the browser is defined and controlled using the Angular framework.
• **Angular is an MVC framework** where the model is made up of JSON or JavaScript objects, the view is HTML/CSS, and the controller is made up of Angular JavaScript.


Browser
Angular
JavaScript Client Side
JSON/XML/ etc.
HTML/CSS/ Images
User Interactions
User
HTML
CSS
HTTP GET/PUT/AJAX
JSON
JPG
JS
Server
Node.js
Express Webserver
Server-Side Scripts
Other Services
MongoDB
Files

The role of Express

- The Express module acts as the webserver in the Node.js-to-Angular stack.
- The Express module extends Node.js to provide several key components for handling web requests.
- For example, the Express module provides the ability to easily set up destination routes (URLs) for users to connect to.
- It also provides great functionality on working with the HTTP request and response objects.
- It relies on other modules called *middleware* to provide the functionality that most applications will need.

Features of Express

Route management: Express makes it easy to define routes (URL endpoints) that tie directly to Node.js script functionality on the server.

Error handling: Express provides built-in error handling for documents not found and other errors.

Easy integration: An Express server can easily be implemented behind an existing reverse proxy system such as Nginx or Varnish. This allows it to be easily integrated into your existing secured system.

Cookies: Express provides easy cookie management.

Session and cache management: Express also enables session management and cache management.

• At the heart of Express is a router, which essentially takes a client request, matches it against any routes that are present, and executes the handler function that is associated with that route.
• The handler function is expected to generate the appropriate response.
• A route specification consists of an HTTP method (GET, POST, etc.), a path specification that matches the request URI, and the route handler.
• Express application - created using the express() function.
• Middleware is installed for handling static files.
• A middleware function deals with any request matching the path specification, regardless of the HTTP method.

Routing: When a request is received, the first thing that Express does is match the request to one of the routes. The request method is matched against the route's method.

Route parameters are named segments in the path specification that match a part of the URL.

Route Lookup: Multiple routes can be set up to match different URLs and patterns. The router does not try to find the best match; instead, it tries to match all routes in the order in which they are installed. The first match is used.

Once a route is matched, the handler function is called, which was an anonymous function supplied to the route setup function.

- The parameters passed to the handler are a request object and a response object. The handler function does not return any value. But it can inspect the request object and send out a response.
- Express is a web framework that has minimal functionality of its own. An Express application is essentially a series of middleware function calls.

Angular

- Angular is a client-side framework developed by Google.
- It provides all the functionality needed to handle user input in the browser, manipulate data on the client side, and control how elements are displayed in the browser view.
- It is written using TypeScript. The entire theory behind Angular is to provide a framework that makes it easy to implement web applications using the MVC framework.

Benefits of Angular

Data binding: Angular has a clean method to bind data to HTML elements using its powerful scope mechanism.

Extensibility: The Angular architecture allows you to easily extend almost every aspect of the language to provide your own custom implementations.

Clean: Angular forces you to write clean, logical code.

Reusable code: The combination of extensibility and clean code makes it easy to write reusable code in Angular.

Support: Google is investing a lot into this project, which gives it an advantage over other similar initiatives.

Compatibility: Angular is based on TypeScript, which makes it easier to begin integrating Angular into your environment and to reuse pieces of your existing code within the structure of the Angular framework.

Node.js

Node.js is a development framework based on Google's V8 JavaScript engine and it's not a programming language.

Node.js is an open-source, cross-platform runtime environment for executing JavaScript code outside a browser.

Therefore, Node.js code is written in JavaScript and then compiled into machine code by V8 to be executed.

Node.js is just JavaScript, so you can easily take functionality from a client-side script and place it in a server-side script.

Also, the webserver can run directly within the Node.js platform as a Node.js module, so it makes it much easier than, Apache at wiring up new services or server-side scripts.

The following are just a few reasons why Node.js is a great framework to start from:

JavaScript end-to-end: One of the biggest advantages to Node.js is that it allows you to write both server- and client-side scripts in JavaScript. There have always been difficulties in deciding where to put scripting logic. Too much on the client side makes the client cumbersome and unwieldy, but too much on the server side slows down web applications and puts a heavy burden on the webserver.

Event-driven scalability: Node.js applies a different logic to handling web requests. Rather than having multiple threads waiting to process web requests, they are processed on the same thread using a basic event model. This allows Node.js webservers to scale in ways that traditional webservers never can.

Extensibility: Node.js has a great following and an active development community. New modules to extend Node.js functionality are being developed all the time. Also it is simple to install and include new modules in Node.js, making it easy to extend a Node.js project to include new functionality in minutes.

Time: Time is valuable. Node.js is super easy to set up and develop in. In only a few minutes, you can install Node.js and have a working webserver.

MongoDB

• MongoDB is an agile and scalable NoSQL database.
The name Mongo comes from “hu**mongo**us.”

• It is based on the NoSQL document store model.

• MongoDB provides great website backend storage for high traffic websites that need to store data such as user comments, blogs, or other items because it is fast, scalable and easy to implement.

• Node.js supports a variety of DB access drivers, so the data store could just as easily be MySQL or some other database.

However, the following are some of the reasons that MongoDB really fits in the Node.js stack well:

• **Document orientation:** MongoDB is document-oriented, the data is stored in the database in a format that supports both server-side and clientside scripts.

High performance: MongoDB is one of the highest-performing databases available. Especially today when more and more people interact with websites, it is important to have a backend that can support heavy traffic.

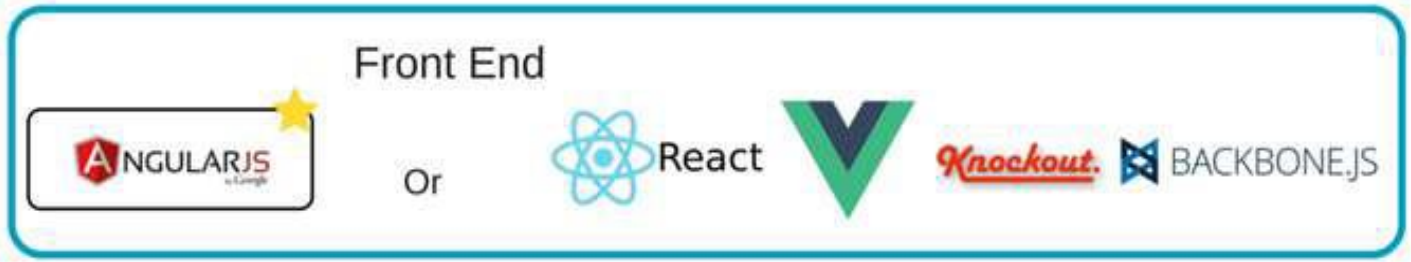
• **High availability:** MongoDB's replication model makes it easy to maintain scalability while keeping high performance.

• **High scalability:** MongoDB's structure makes it easy to scale horizontally by
sharing the data across multiple servers.

• **No SQL injection:** MongoDB is not susceptible to SQL injection, because objects are stored as objects, not using SQL strings.

React

- React is an open-source JavaScript library maintained by Facebook that can be used for creating views rendered in HTML.
- Unlike AngularJS, React is not a framework. It is a library.
- Not just Facebook itself, but there are many other companies that use React in production like Airbnb, Atlassian, Bitbucket, Disqus, Walmart, etc
- It is a component-based front-end library responsible only for the view layer of an MVC (Model View Controller) architecture.
- React is used to create modular user interfaces and it promotes the development of reusable UI components that display dynamic data.


Front End
ANGULARJS
Or
React
Knockout.
BACKBONE.JS


Back End
node
express
Or
METEOR
sails
restify
KeystoneJS


Database
mongoDB
Or
MySQL
PostgreSQL
CouchDB
cassandra

Benefits of using web development framework

- Speed up application development, reduce errors, write less code, reuse code, simplify debugging and increase reliability.
- Used and stick faster to industry standards.
- Better security

Questions

1. Web Application Framework
2. GET request
3. POST request
4. AJAX
5. Example of MVC framework
6. Webserver
7. Node.js is written in???
8. MongoDB based on???
9. React