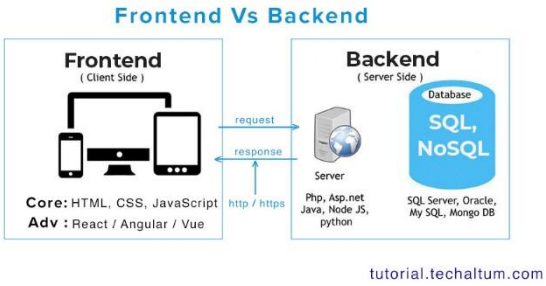
MEARN STACK ASSIGNMENT 1

1. What is Web Development?

Web development is the process of **creating, building, and maintaining websites or web applications** that run on the internet or an intranet.
It involves everything from a **simple static webpage** to **complex dynamic web apps** like e-commerce sites, social media platforms, and dashboards.

Web development = **Frontend (what users see)** + **Backend (how it works)** + **Database (where data is stored)**



1. Frontend Development (Client Side)

Frontend is everything the **user can see and interact with**.

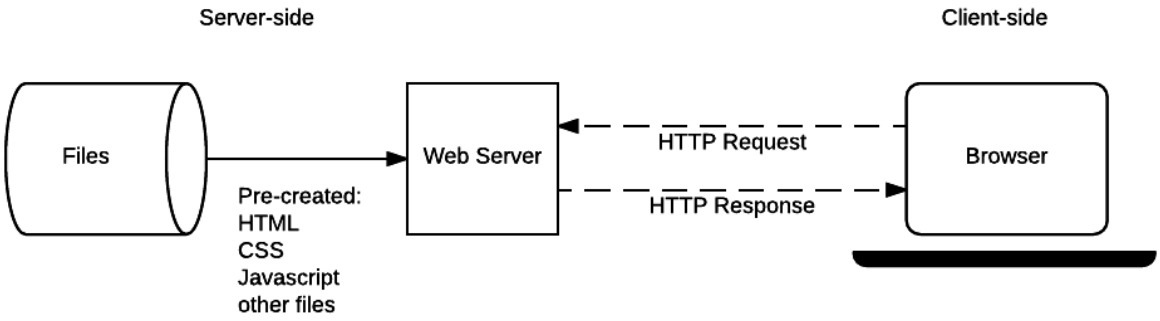
Technologies Used:

- **HTML** – Structure of the webpage

- **CSS** – Styling, layout, colors, fonts
- **JavaScript** – Interactivity (buttons, forms, animations)

2. Backend Development (Server Side)

Backend is the **brain** of the website.
Users don't see it, but it controls everything.



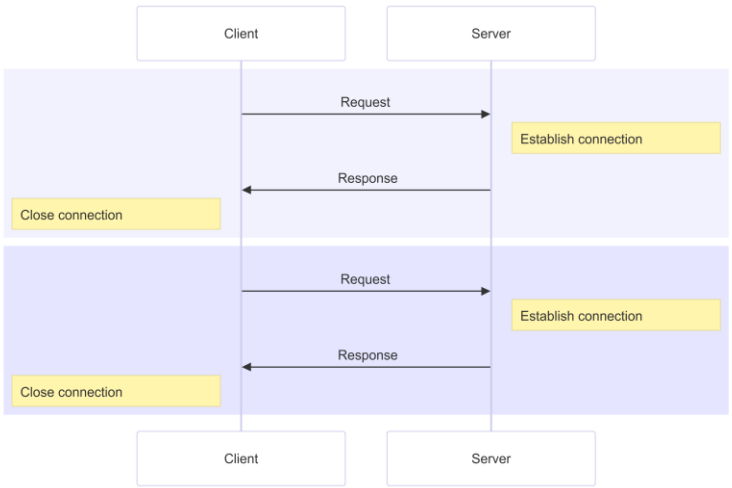
Backend Technologies:

- Languages: Java, Python, PHP, JavaScript (Node.js)
- Frameworks: Spring Boot, Django, Express

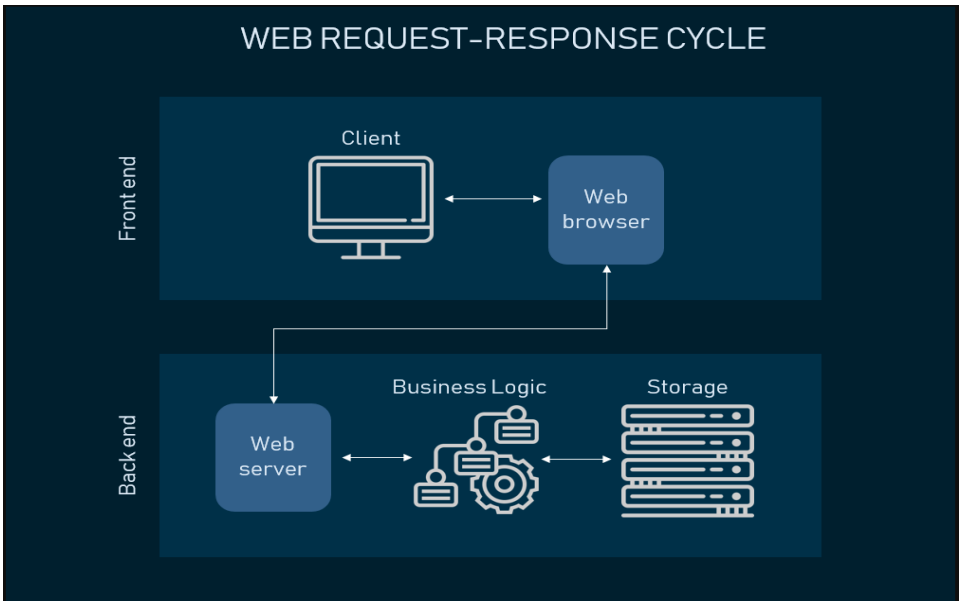
3. Database (Data Storage)

Database is where we stores data **permanently**. And during any processing required data is fetched from database when it triggers

Workflow



2. Difference between frontend and backend with examples.



In **web development**, a website or web application is divided into **two major parts**:

1. **Frontend** – what the user sees and interacts with
2. **Backend** – how the system works behind the scenes

Both parts work together to make a **complete and functional website**.

Frontend Development (Client-Side)

Frontend development refers to the part of a website that is **visible to users** and runs in the **web browser**.

It focuses on **user interface (UI)** and **user experience (UX)**.

Responsibilities of Frontend

- Designing page layout
- Displaying text, images, and videos
- Creating buttons, forms, and menus
- Handling user interactions (clicks, input)
- Making website responsive (mobile, tablet, desktop)

Technologies Used in Frontend

HTML, CSS , JavaScript , React/Angular

Backend Development (Server-Side)

Backend development refers to the part of a website that **runs on the server** and is **not visible to users**.

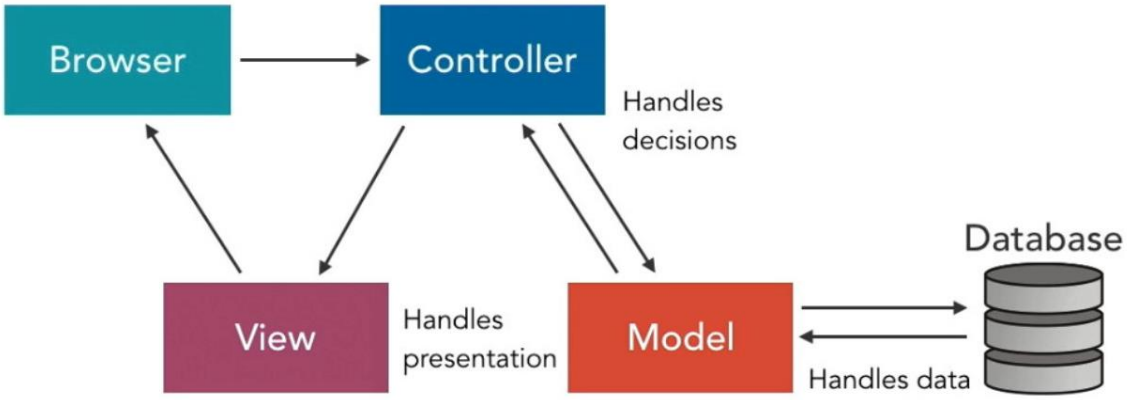
It controls **logic, security, and data processing**.

Responsibilities of Backend

- User authentication and authorization
- Processing requests from frontend
- Business logic implementation
- Database operations (insert, update, delete)
- Ensuring security and performance

Technologies Used in Backend

Java / Python / PHP, Node.js, Spring Boot / Django, MySQL / MongoDB

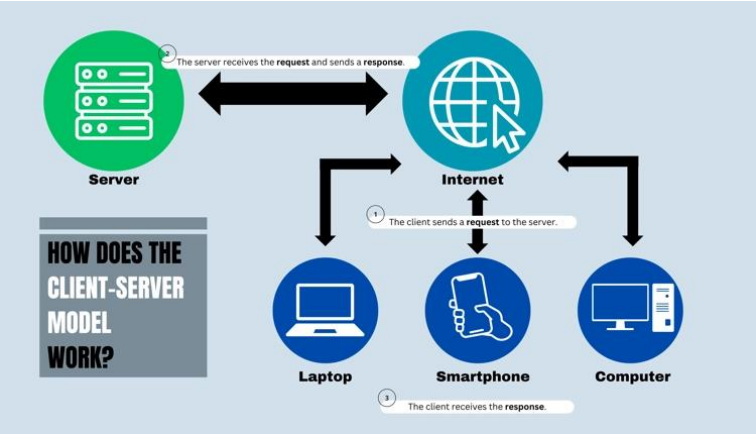


---

3. Explain client–server communication with a neat diagram.

Client–server communication is a model in which a **client** (user's device or browser) sends a **request** to a **server**, and the **server processes the request and sends back a response**.

This communication is the **foundation of web applications**, online services, email systems, and cloud computing.



1. Client

- A client is a **user-side application**
- Examples: Web browser (Chrome), mobile app
- Role: **Requests services or data**

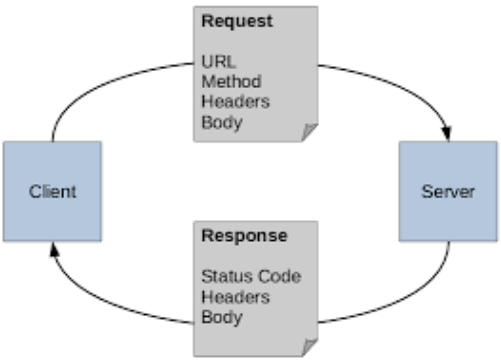
2. Server

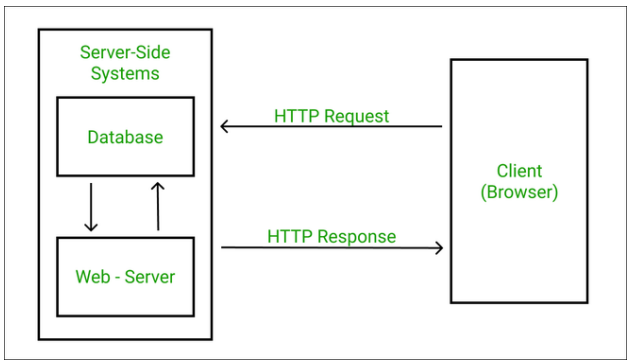
- A server is a **powerful system** that provides services
- Examples: Web server, application server
- Role: **Processes requests and sends responses**

3. Network

- Medium through which data is transferred
- Usually the **Internet**
- Uses protocols like **HTTP/HTTPS**

Working of Client–Server Communication





Step 1: Client Sends Request

Step 2: Server Receives Request

Step 3: Server Processes Request

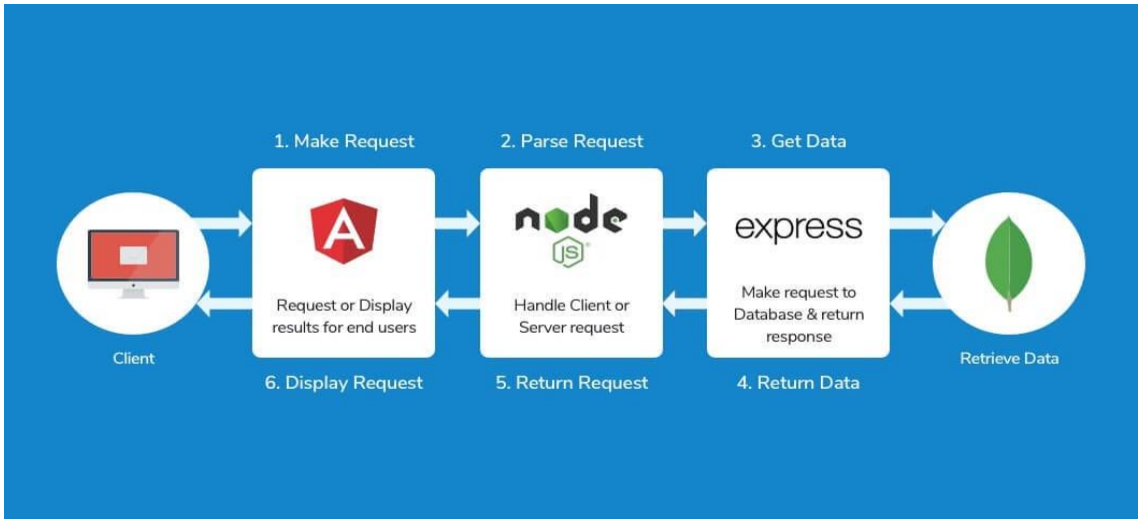
Step 4: Server Sends Response

Step 5: Client Displays Response

---

4. What is MEAN stack?

MEAN Stack is a **full-stack web development technology stack** used to build **dynamic, scalable, and modern web applications** using **JavaScript at every layer** of the application.



1. M-MongoDB (Database Layer)

- A **NoSQL database**
- Stores data in **JSON-like documents**
- Flexible and scalable
- Ideal for handling large amounts of data

2. E-Express.js (Backend Framework)

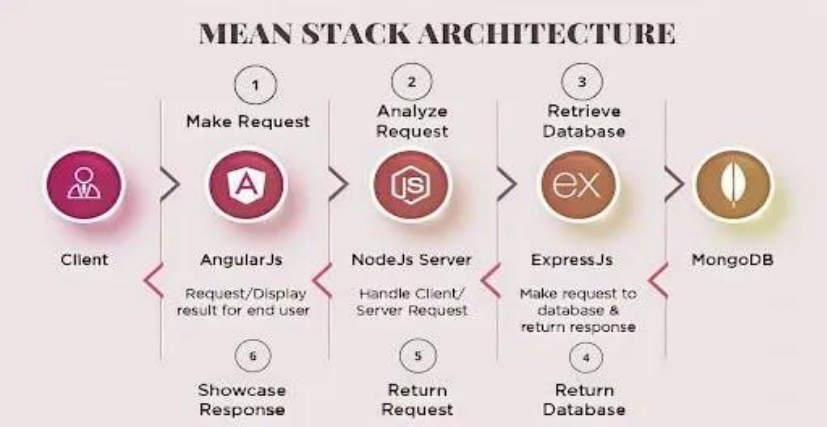
- A **lightweight backend framework** built on Node.js
- Simplifies server-side coding
- Handles routing, middleware, and APIs

3. A-Angular (Frontend Framework)

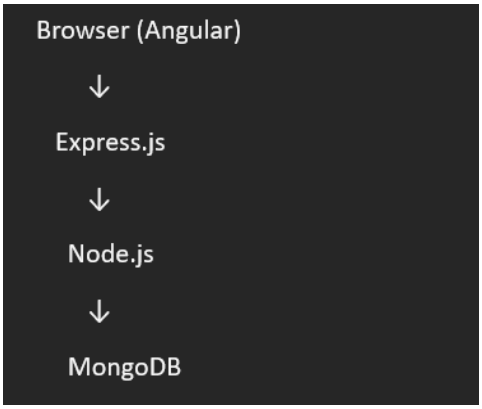
- A **frontend framework** used to build **single-page applications (SPA)**
- Runs in the browser
- Provides dynamic and interactive UI
- Communicates with backend using APIs

4. N-Node.js (Runtime Environment)

- JavaScript runtime that runs on the **server**
- Uses non-blocking, event-driven architecture
- Highly scalable and fast



1. User interacts with **Angular frontend**
2. Angular sends request to **Express server**
3. Express runs on **Node.js**
4. Data is fetched/stored in **MongoDB**
5. Response sent back to Angular
6. UI updates on browser



5. Install Angular CLI using npm install -g @angular/cli

Angular CLI (Command Line Interface) is a tool used to:

- Create Angular projects
- Generate components, services, modules
- Build, run, and test Angular applications easily

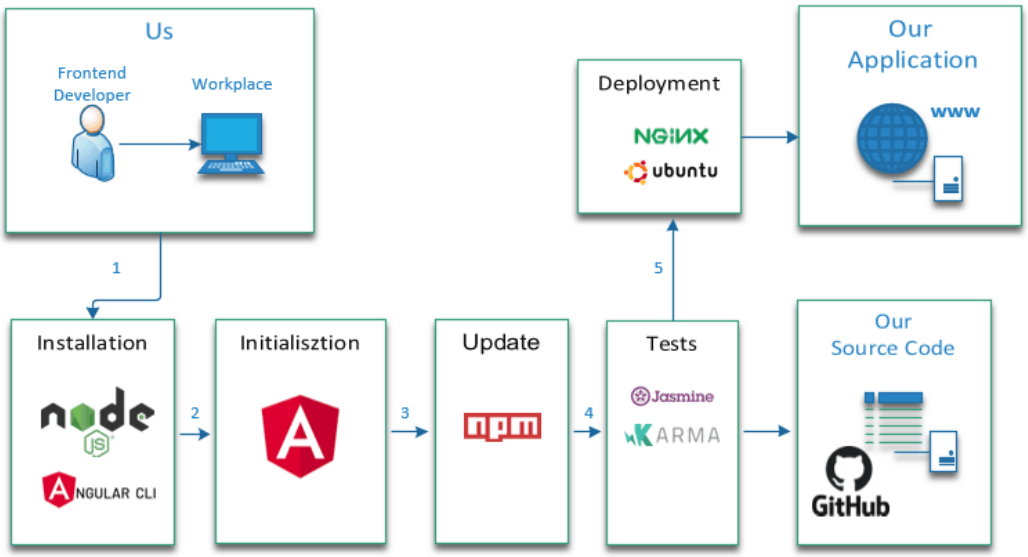
  It saves time by automating repetitive tasks.

Before installing Angular CLI, the system must have: **Node.js**

- Angular CLI works on **Node.js**
- Node.js includes **npm (Node Package Manager)**

  Used to install Angular CLI (comes with Node.js

Installation Process



Step 1: Open Command Prompt / Terminal

- Windows → Command Prompt / PowerShell

Step 2: Install Angular CLI Globally

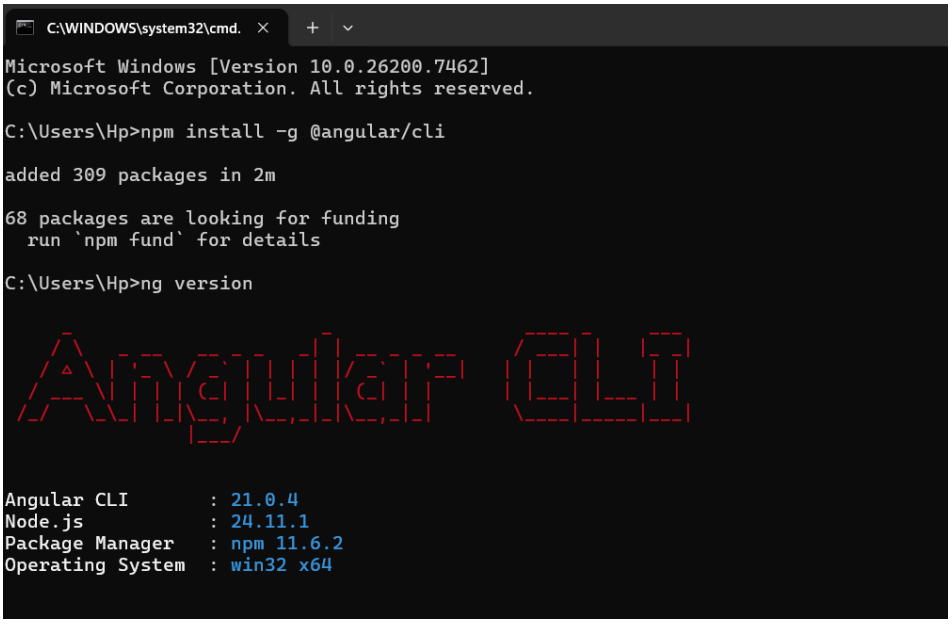
Run the command:

npm install -g @angular/cli

To Globally install and access from everywhere, allows you to use ng command anywhere.

Step 3: Verify Installation

ng version



What Happens Internally?

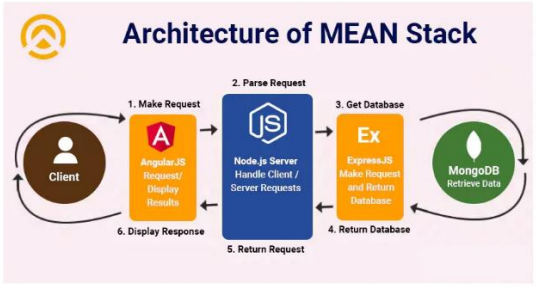
1. npm downloads Angular CLI package
2. Files are installed globally on system
3. ng command becomes available
4. CLI can now create and manage Angular apps

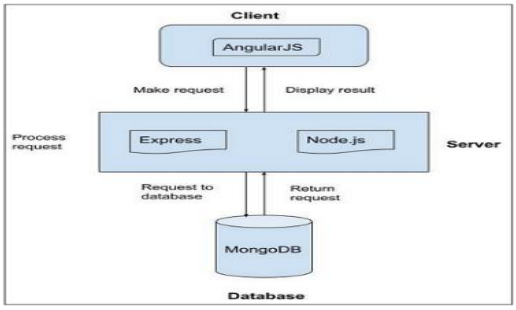
Open in browser:

http://localhost:4200

---

6. Draw the MEAN architecture workflow: Client (Angular) → Server (Express + Node.js) → Database (MongoDB)





MEAN Stack Architecture Workflow

1. Client – Angular

- Runs in the user's **web browser**
- Displays UI (forms, buttons, pages)
- Sends **HTTP requests** (GET, POST, PUT, DELETE)
- Receives response and updates UI
- For User Interface

2. Server – Express + Node.js

- Express handles **routing and APIs**
- Node.js executes backend JavaScript
- Processes client request
- Applies business logic

- Communicates with database

3. Database – MongoDB

- Stores data in **JSON-like documents**
- Performs CRUD operations
- Sends data back to server

4. Response Flow

- MongoDB → sends data to server
- Server → sends JSON response
- Angular → displays result to user