1. C’est quoi la MERN Stack ?

MERN est un ensemble de 4 technologies pour créer une application web **complète** (front + back) :

- **M** = **MongoDB** : base de données (backend)
- **E** = **Express.js** : serveur backend
- **R** = **React.js** : interface utilisateur (frontend)
- **N** = **Node.js** : exécute le backend avec JavaScript

MERN c’est une combinaison pour faire un site de A à Z.

MERN Stack t5allik t3mel site web kamel, front + back, b JavaScript.

2. Explication des composants de MERN

◆ **MongoDB**

Une base de données où on enregistre les infos (utilisateurs, messages, produits…)

(T7ot fiha data mte3ek comme fichiers JSON.)

◆ **Express.js**

Crée un serveur backend qui répond aux requêtes (API).

(Houa li ykoun bin frontend w MongoDB, yjib w yb3ath données.)

◆ **React.js**

Crée le design et les composants visibles du site (boutons, formulaires, pages…).

(Ykhallik t3mel interface.)

◆ **Node.js**

Permet d'utiliser JavaScript côté serveur.

(Ykhallik tkoun backend JavaScript)

3. Architecture MVC (Model - View - Controller)

◆ **Model (Modèle)**

Représente les données (ex : utilisateur, article…). C'est la **structure** des infos.

(Fih les schemas mte3 MongoDB. Exemple : nom, email, password…)

◆ **View (Vue)**

Ce que l'utilisateur voit : les pages, boutons, formulaires (côté React).

(Taffichi les données lel user. Exemple : liste des produits.)

◆ **Controller**

C'est la **logique** : il récupère les données du Model et les envoie à la View.

(Yjib data men Mongo, yformilha, w yb3athha lel frontend.)

➡ MVC t9assem l'application 3 parties pour organiser w traya7 rouhek.

4. Les packages Back-end (Node + Express)

- **axios** Pour envoyer des requêtes HTTP (get, post...) entre le front et le back.
  (Taba3th w tjib données bin frontend w backend.)
- **bcrypt** Pour crypter les mots de passe avant de les enregistrer.
  (T5alli password ma yet9arach. Ywalli crypté.)
- **concurrently** Pour démarrer frontend et backend en même temps avec une seule commande.
  (React w Node ykhdmou b3adhhom.)
- **cors** Pour permettre au frontend (localhost:3000) de communiquer avec le backend (localhost:5000).
  (Ykhalihom y7kiw ensemble même si adresses différents.)
- **dotenv** Pour utiliser un fichier .env qui contient les informations secrètes (clé, mot de passe, URL…).
  (T7ot clé privée fih, ma yetnasharch f GitHub.)
- **express** Pour créer le serveur backend avec Node.js.
  (Ykhallek t3ayet routes, tnaffeth API, etc.)
- **express-validator** Pour valider les données des formulaires (vérifier si email est bon, mot de passe assez long, etc.)
  (Y9ollek l'input mte3ek mrigla wala la.)
- **jsonwebtoken (jwt)** Pour gérer les connexions sécurisées avec JWT.
  (Ki user yconnecti, ya5ou token bach yb9a connecté.)
- **mongoose** Pour gérer la base MongoDB facilement avec des modèles.
  (Tkteb schema w yb9a t5alli/tjib données bch ykoun 7aja tertiba.)

- **multer** Pour envoyer et stocker des fichiers (ex: images) dans le backend.
  (Tuploadi image ou fichier men formulaire.)
- **nodemon** Pour redémarrer automatiquement le serveur quand tu modifies ton code.
  (Au lieu de relancer manuellement, houwa yrefresh wa7dou.)
- **passport** Pour gérer l'authentification des utilisateurs.
  (Tconnecti user en toute sécurité.)
- **passport-jwt** Une extension de passport pour gérer les connexions avec les tokens JWT.
  (Yekhdem m3a jsonwebtoken pour vérifier le token.)
- **socket.io** Pour faire de la communication en temps réel (ex: chat, notifications en direct…).
  (Ykhallek t3mel messagerie instantanée b React + Node.)

kol package fih fonction spécifique, kenek t7eb ta3ref kif tconnecti, ta3ref user, ta3ref chat... kol wa7ed 3andou khidma.

5. Les packages Front-end (React)

- **@emailjs/browser** : envoyer email directement depuis le navigateur
- **@headlessui/react** : composants accessibles sans style (menu, modal...)
- **@heroicons/react** : icônes modernes à intégrer dans React
- **@reduxjs/toolkit** : meilleure façon d'utiliser Redux (stock global)
- **react-redux** : connecter React à Redux
- **react-router-dom** : navigation entre pages

- **react-simple-chatbot** : créer un chatbot (robot qui parle)
- **react-awesome-reveal** : animations d'apparition
- **react-icons** : bibliothèque d'icônes
- **socket.io-client** : communication temps réel avec backend
- **sweetalert2** : alertes jolies et animées
- **web-vitals** : mesurer performance de l'app
- **@testing-library/** : tester les composants React
- **react / react-dom / react-scripts** : coeur de l'application React

kol package f React yjib animation, chat, email, icons... bsh tzayen l'app w tzidha fonctions utiles.

Souviens-toi :

Back = Express + Mongo + Node

Front = React

MVC = Organisation (données, logique, affichage)

6. Principe de l'authentification avec JWT (JSON Web Token)

JWT = JSON Web Token

C’est une **clé sécurisée** que le serveur donne au client quand il se connecte.

(Ki l’utilisateur yconnecti b email/password, l’backend y9oulou "ok", w y3tih token.)

Comment ça marche ?

1. Le client (React) envoie email + mot de passe ➞ au backend (Express).
2. Le backend vérifie dans MongoDB si les infos sont correctes.
3. Si c’est bon, il crée un **JWT** et le renvoie au client.
4. Le client garde ce token (en localStorage).
5. À chaque requête suivante, le client envoie ce **token** dans les headers.
6. Le serveur vérifie le token ✚ autorise ou refuse l’accès.

Que contient un JWT ?

Un token a 3 parties :

- **Header** (type + algorithme)
- **Payload** (données, ex: id utilisateur, rôle)
- **Signature** (clé secrète du serveur)

l’utilisateur ya5ou carte (token) men l’backend, w ywalli ywarriha fi kol action y3melha. Si la carte est valide, ykammel, sinon y’out.