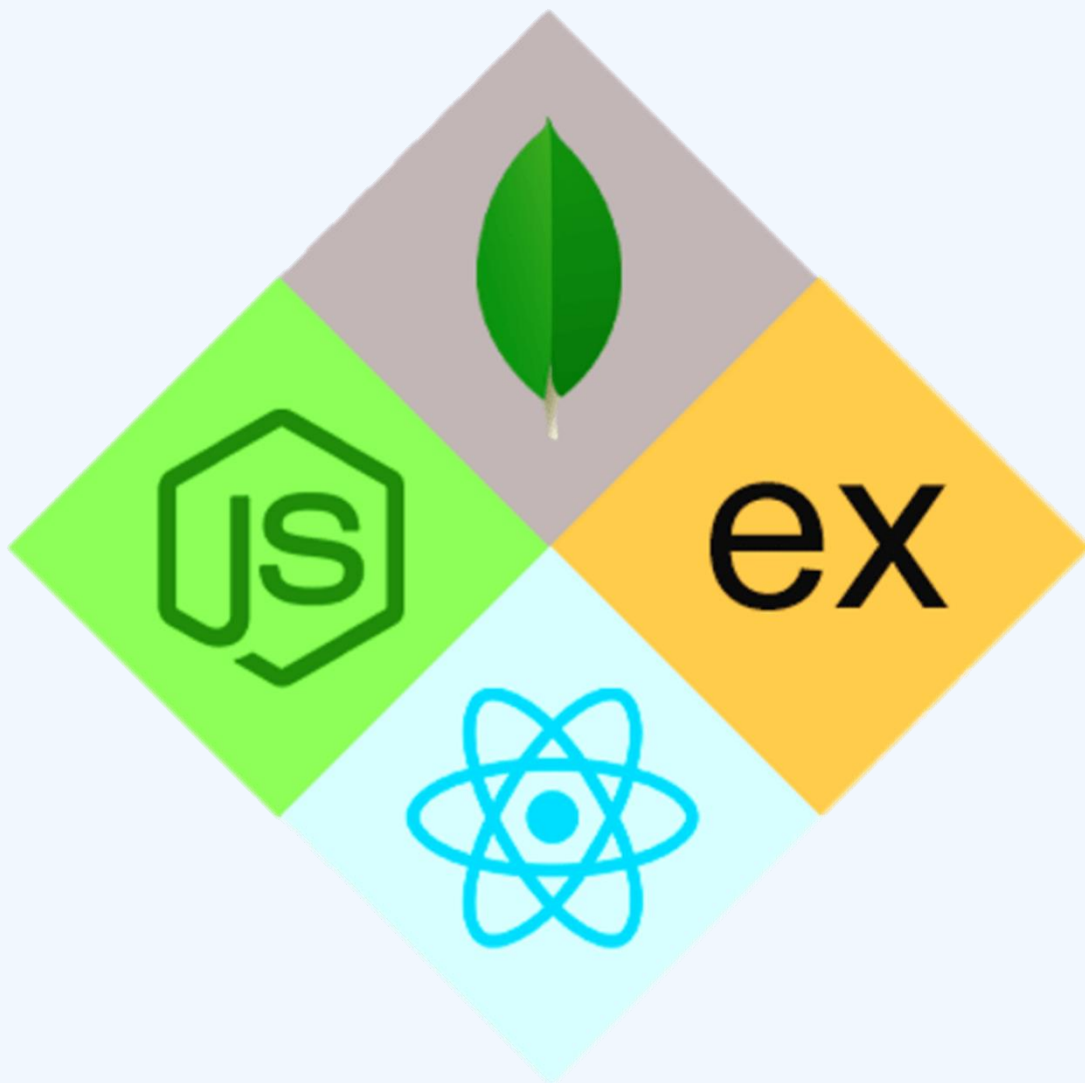


# The Ultimate Full (MERN) Stack Roadmap



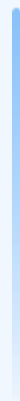
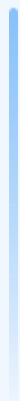
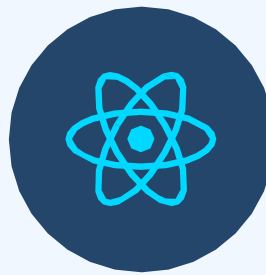
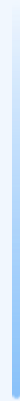
# MongoDB

Document database



# React.js

Client-side JS framework



# Express.js

Node.js web framework

# Node.js

JavaScript web server

# What is MERN Stack?

MERN Stack is a Javascript Stack that is used for easier and faster deployment of full-stack web applications.

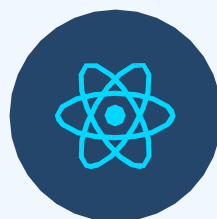
MERN Stack comprises of 4 technologies namely:

**MongoDB**, **Express**, **React** and **Node.js**. It is designed to make the development process smoother and easier

The MERN architecture allows you to easily construct a 3-tier architecture (frontend, backend, database) entirely using JavaScript and JSON

Using these four technologies you can create absolutely any application that you can think of everything that exists in this world today.

Now let's understand each technology one by one.





# MongoDB

# MERN

MongoDB forms the **M** of the **MERN** stack and works pretty well with the JavaScript ecosystem.

MongoDB is a NoSQL database in which data is stored in documents that consist of key-value pairs, sharing a lot of resemblance to JSON.

The data is not stored in the form of tables and that's how it differs from other database programs.

This is how the data stored in MongoDB looks like:

```
{
  _id: ObjectId("2af3b3a615571a1015d782e9")
  name: "JavaScript Mastery"
  email: "jsm@jsmasterypro.com"
  password: "useStrongPasswords!"
  createdAt: 2022-04-04T16:02:32.146+00:00
  updatedAt: 2022-04-05T08:06:10.425+00:00
  __v: 0
}
```



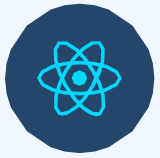
# Express JS

# MERN

Express is a flexible and clean Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node based web applications.

Express helps build the backend very easily while staying in JavaScript ecosystem. It is preferred for self-projects as it helps focus on learning development and building projects pretty fast.

In MERN stack, Express will be used as backend API server which interacts with MongoDB database to serve data to client (React) application.



# React JS

# MERN

React is an open-source JavaScript library that is used for building user interfaces specifically for single-page applications. It's used for handling the view layer for web and mobile apps.

React lets you build up complex interfaces through simple Components, connect them to data on your backend server, and render them as HTML.

Almost all the modern tech companies from early-stage startups to the biggest tech companies like Microsoft and Facebook use React.

The prime reason why react is used, is for Single Page Applications(SPA). SPA means rendering the entire website on one page rather than different pages of the websites.



# Node JS

# MERN

NodeJS is a cross-platform JavaScript runtime environment, it's built on Chrome's V8 engine to run JavaScript code outside the browser, for easily building fast and scalable applications.

The main purpose of NodeJS is simple, it allows us to write our backend in JavaScript, saving us the trouble of learning a new programming language capable of running the backend.

Node.js is the platform for the application layer (logic layer). This will not be visible to the clients. This is where client applications (React) will make requests for data or webpages.

# MERN Roadmap

Now let's dive into what you need to learn in MERN.

First and foremost before you dive into any advanced topics, you first need to learn the core of the web i.e. HTML, CSS, and JavaScript.

## HTML

HTML provides the basic structure.

## CSS

CSS provides the skin to the structure in the form of design, formatting, and layout.

## JavaScript

JavaScript adds interactivity and logic to the website.



# Important things to learn



## HTML

- ✓ Basics
- ✓ Different Tags
- ✓ Forms
- ✓ Semantic HTML
- ✓ SEO Basics
- ✓ Accessibility
- ✓ Best practices

# Important things to learn



## CSS

- ✓ Basics
- ✓ Selectors
- ✓ Positioning
- ✓ Box Model
- ✓ Specificity
- ✓ Media Queries
- ✓ Best practices
- ✓ FlexBox
- ✓ Grid
- ✓ Pseudo Elements
- ✓ Pseudo Classes
- ✓ Animations
- ✓ Transitions
- ✓ Responsiveness

# Important things to learn

## JavaScript

- ✓ Basics
- ✓ DOM
- ✓ Fetch API
- ✓ Async Await
- ✓ Event Listeners
- ✓ ES6+ features
- ✓ Best practices
- ✓ Promises
- ✓ Classes
- ✓ Array Methods
- ✓ String Methods
- ✓ Scoping
- ✓ Hoisting
- ✓ Closures

# MERN Roadmap

Once you know enough front-end development, where you can build great projects, then you should consider diving into learning React.js.

You might be wondering, what are the prerequisites to learn such a great JavaScript library?

There's only one prerequisite and that is – [JavaScript](#).

Keep in mind, do not directly jump into learning React.js, before learning JavaScript, you should be confident with JavaScript because JavaScript is the main thing you'll be working with.

Most importantly, you must understand

- ✓ Syntax
- ✓ ES6+ features

# MERN Roadmap

- ✓ Arrow functions
- ✓ Template literals
- ✓ Array Methods
- ✓ Object property shorthand
- ✓ Destructuring
- ✓ Rest operator
- ✓ Spread operator
- ✓ Promises
- ✓ Async/Await syntax
- ✓ Import and export syntax

# MERN Roadmap



## *Basic things to learn in React.js*

- ✓ File & Folder structure
- ✓ Components
- ✓ JSX
- ✓ Props
- ✓ State
- ✓ Events
- ✓ Styling
- ✓ Conditional Rendering

# MERN Roadmap



*Learn about React.js Hooks –  
the essential hooks to learn:*

- ✓ useState
- ✓ useEffect
- ✓ useRef
- ✓ useContext
- ✓ useReducer
- ✓ useMemo
- ✓ useCallback

# MERN Roadmap



*Also learn these essential things:*








- ✓ Fetching data from APIs
- ✓ Routing
- ✓ Context API
- ✓ Learn to create custom hooks
- ✓ Handling form submits
- ✓ Use cases of less common hooks
- ✓ Higher-Order Components
- ✓ React DevTools



# MERN Roadmap



*Then learn some of the React.js  
UI Frameworks*

- ★  Material UI
- ★  Ant Design
- ★  Chakra UI
-  React Bootstrap
-  Rebass
-  Blueprint
-  Semantic UI React

# MERN Roadmap



*Learn to use some of the most popular React.js packages*



React Router



Axios



Styled Components



React Hook Form



React Query



Storybook



Framer Motion

# MERN Roadmap



*Learn how to manage state  
with state management tools*



Redux



MobX



Hookstate



Recoil



Akita

# MERN Roadmap



*Learn to test your apps with some of these libraries/frameworks*



Jest



Testing Library



Cypress



Enzyme



Jasmine



Mocha

# MERN Roadmap

Before you dive into backend, you should first consider learning or atleast understanding some of these concepts mentioned below:

- ✓ HTTP/HTTPS
- ✓ RESTful APIs
- ✓ CRUD
- ✓ CORS
- ✓ JSON
- ✓ Package Manager
- ✓ MVC architecture
- ✓ GraphQL

# MERN Roadmap



Now if you feel confident with React.js & concepts mentioned, you can jump into learning [Node.js](#).

As mentioned earlier, NodeJS is a cross-platform JavaScript runtime environment that means we can use JavaScript on the server.

Isn't that amazing? With the help of Node.js we don't have to learn any other programming language. We already know one: **JavaScript**.

Now let's take a look at, what you need to learn in Node.js, well there's not much specific to NodeJS that you have to learn to build a MERN stack application, here are some related things you should take a look at

## ✓ Initialising a npm package

# MERN Roadmap



- ✓ Installing npm packages through npm or yarn
- ✓ Understanding the package.json file
- ✓ Create a basic http server in Node.js
- ✓ Importing and exports modules
- ✓ Working with FileSystem in Node.js
- ✓ HTTP Protocols
- ✓ Events & Event Emitters
- ✓ Global object

# MERN Roadmap



After you know the basics of Node.js you can start learning [Express.js](#).

It's the most popular web application framework which uses NodeJS. In MERN stack applications, Express's role is to manage our backend API server.

Express is used to listen to a particular port of our server for requests from the client or the frontend. We can create different routes for each endpoint.

For example:

```
GET http://domain.com/blogs    → Fetches all blogs
GET https://domain.com/blog/1  → Fetches the blog with ID of 1
```



# MERN Roadmap



A simple example of Express.js Server.

```
const express = require('express');
const app = express();

app.get('/', (req, res) => {
  res.json({ message: "Hello World!" });
});

app.listen(5000);
```

Now, let's dive into things that you should learn concerning Express, as a MERN stack developer:

- ✓ Basic server in Express
- ✓ Creating & handling routes

# MERN Roadmap



- ✓ Learn Middlewares
- ✓ CRUD operations
- ✓ Error Handling
- ✓ Using MVC Architecture
- ✓ Authentication & OAuth
- ✓ API Versioning
- ✓ Rate Limiting
- ✓ Creating custom middlewares

Reading JSON form data sent

- ✓ from frontend via `express.json()` middleware

# MERN Roadmap



Once you know how to use Express.js and you'll start creating projects, you'll notice that you need some kind of database to store data of your applications, data such as (*user profiles, content, comments, uploads, etc.*)

And that's where **MongoDB** comes into play. We use MongoDB because it's the most popular one, and it works well in the JavaScript ecosystem, because it's just similar to JSON.

JSON documents created in your React.js front end can be sent to the Express.js server, where they can be processed and (assuming they're valid) stored directly in MongoDB for later retrieval.

Concepts to learn in MongoDB are:

# MERN Roadmap



- ✓ SQL Vs NoSQL
- ✓ MongoDB database structure
- ✓ Setting up local MongoDB or cloud MongoDB Atlas database
- ✓ Perform CRUD (*Create, Read, Update & Delete*) operations on the database
- ✓ Indexing
- ✓ Aggregations
- ✓ Ad-hoc query
- ✓ Creating models and schemas

# MERN Roadmap



To interact with MongoDB better, we generally use an ODM or Object Data Modelling library like [Mongoose](#).

Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js. It manages relationships between data, provides schema validation, and is used to translate between objects in code and the representation of those objects in MongoDB.

Important things to learn in Mongoose:

- ✓ Defining the Schema
- ✓ Mongoose data validation
- ✓ Understanding mongoose pre and post hooks

# MERN Roadmap

Also there are some essential things you should learn to become a fantastic MERN stack developer.



Git



GitHub



Terminal (CLI)



Postman



Payment Gateways



Testing

# MERN Roadmap

## *Learn to deploy your frontend*

Some popular free websites that you can use to deploy your frontend application or client-side code.



Netlify



Vercel



Firebase



Github Pages



Render

# MERN Roadmap

## *Learn to deploy your backend*

The 2 popular free services that you can use to deploy your backend application or server-side code.



Vercel



Heroku



# MERN Roadmap

You can also use any popular cloud service, almost all cloud services offer 1 year free trial, but you need to put your card information. So it's up to you.



Google Cloud



Digital Ocean



AWS



Azure



Linode

# CRUD

CRUD Stands for

**Create Read Update Delete**

In a REST environment, CRUD often corresponds to the HTTP methods GET, POST, PUT/PATCH, and DELETE.

If you think about it, every app is a CRUD app in a way. You can find this pattern everywhere:

## **Building a blog:**

create posts, read them, and delete them.

## **Building a social media network with users**

create users, update user profiles, and delete users

## **Building a movie app**

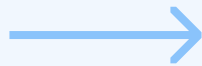
add your favorites, rate, and delete them

# CRUD

In regards to its use in RESTful APIs, CRUD is the standardized use of HTTP Action Verbs.

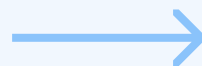
This means that if you want to create a new record you should be using “**POST**”. If you want to read a record, you should be using “**GET**”. To update a record use “**PUT**” or “**PATCH**”. And to delete a record, use “**DELETE**”.

Create



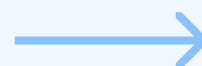
POST

Read



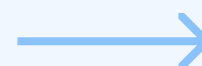
GET

Update



PUT / PATCH

Delete



DELETE

# CRUD

Let's look at some code examples of our Memories application, to understand CRUD better:

```
export const createPost = async (req, res) => {  
  const { title, message, selectedFile, creator, tags } = req.body;  
  
  const newPostMessage = new PostMessage(  
    { title, message, selectedFile, creator, tags }  
  )  
  
  try {  
    await newPostMessage.save();  
    res.status(201).json(newPostMessage );  
  } catch (error) {  
    res.status(409).json({ message: error.message });  
  }  
}
```

## CreatePost:

We tell the createPost function that here is the data object for the new Post, please insert it into the database. Or we can say, we're **creating** a new post.

# CRUD

```
export const getPosts = async (req, res) => {  
  try {  
    const postMessages = await PostMessage.find();  
    res.status(200).json(postMessages);  
  } catch (error) {  
    res.status(404).json({ message: error.message });  
  }  
}
```

## GetPosts

Return all the posts from the database. That means we're **reading** all the post from the database.

```
export const updatePost = async (req, res) => {  
  const { id } = req.params;  
  const { title, message, creator, selectedFile, tags } = req.body;  
  
  const updatedPost =  
    { creator, title, message, tags, selectedFile, _id: id };  
  
  await PostMessage.findByIdAndUpdate(id, updatedPost, { new: true });  
  res.json(updatedPost);  
}
```

## UpdatePosts

Find a post with this ID and then update the post with the new data. So we're **updating** the post.

# CRUD

```
export const deletePost = async (req, res) => {  
  const { id } = req.params;  
  
  if (!mongoose.Types.ObjectId.isValid(id)) {  
    return res.status(404).send(`No post with id: ${id}`);  
  }  
  
  await PostMessage.findByIdAndRemove(id);  
  res.json({ message: "Post deleted successfully." });  
}
```

## DeletePosts

Find a post with this ID and delete it from the database. So here we're **deleting** a post.

While this application may have more complicated database queries included, without all of the basic CRUD operations this app will be nothing.

# MERN Project Ideas



Social Media App



Chat App



E-commerce Platform



Hotel Booking App



Travel Log App



Task Management Tool

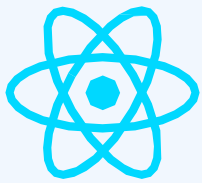


Discord Clone



Bookstore Library

# How it works



User browsing  
React App



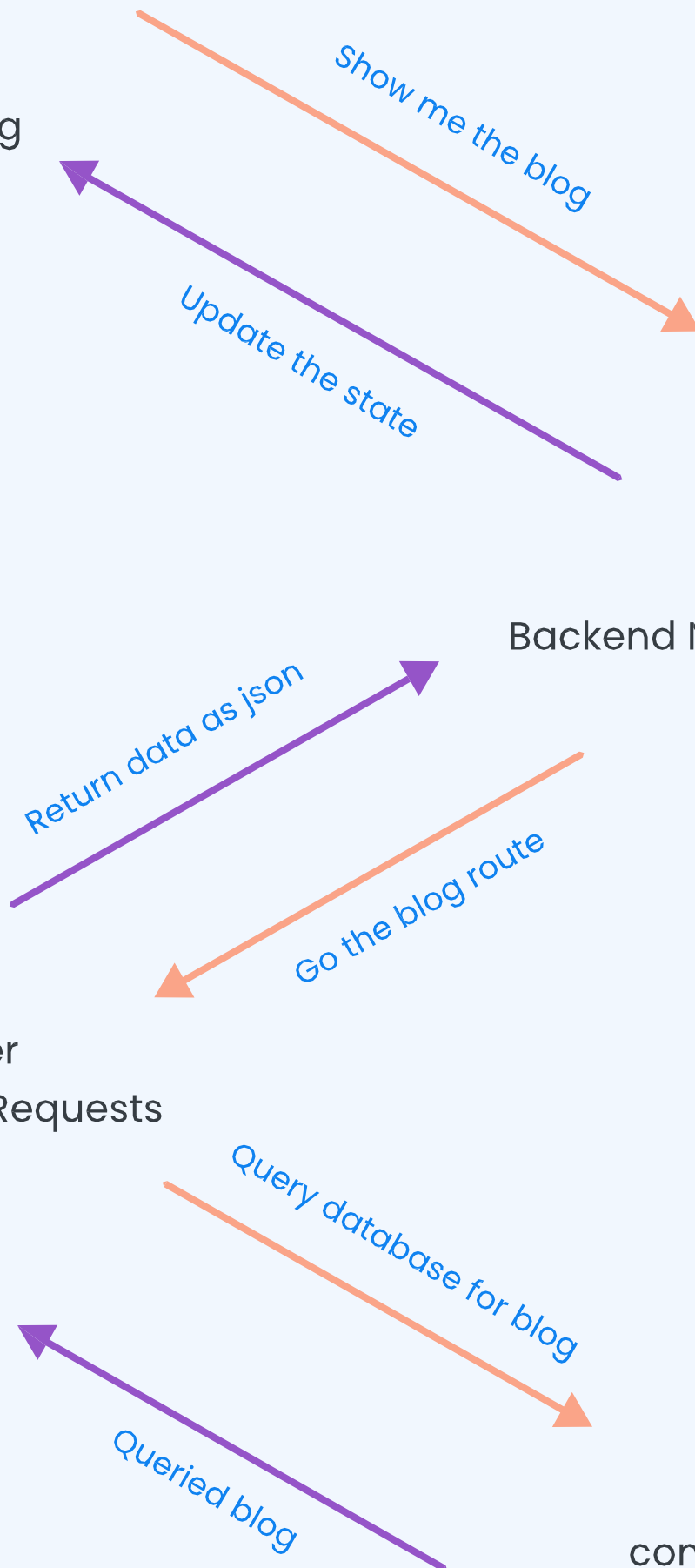
Backend Node.js Code



Express Server  
Listening for Requests



MonogDB  
containing data





# Important Note

You don't need to learn all the things mention in this roadmap to become a MERN stack developer or get a job as a MERN developer.

There is no end of learning in web development there's always something to learn.

**So never stop learning!**



<https://www.linkedin.com/in/saqlainshah/>



**03475484803**



**[linktr.ee/saqlain\\_shah](https://linktr.ee/saqlain_shah)**

**An Adventure from convince toward cognition and consciousness**



Business in the service of knowledge



Let's us introduce You to the World of

# Full Stack | MERN Stack Web Development



express



## Tools & Technologies:

HTML&CSS, SASS, Material-UI, JavaScript, ReactJS, NodeJS, ExpressJS, MongoDB, Mongoose, NPM, Git, Github,

- ❑ Get real experience to develop and learn the highly in-demand skill for the 21st century today!

Enroll Today



**03475484803**



**linktr.ee/saqlain\_shah**