📃 From Static to Stack: Upgrade Your Website to MERN with Vite + React

---

📘 Chapter 1: Getting Ready to Upgrade — From Static to Stack

✅ What You'll Learn

- Understand the difference between static and full-stack sites
- Learn what MERN is and how it works
- Understand why React (with Vite) and GitHub are the starting point

🔍 Why It Matters

Static sites are limited. With MERN, you unlock dynamic content, real data storage, and app-like behavior.

□ Before You Begin

- A working static website (HTML/CSS/JS)
- Node.js and npm installed
- GitHub account created

👥 Step-by-Step

1. Compare static vs full-stack (use a T-chart)
2. Learn MERN:
    - MongoDB: stores data
    - Express: server routes/API
    - React: dynamic frontend

    - Node.js: JS runtime for backend

3. Explore the dev workflow and deployment strategy

💭 Reflect + Extend

- What part of your static site could become dynamic?

🚀 Real-World Task

Pick a site (e.g., portfolio) and map which parts will become dynamic with MERN.

📘 Chapter 2: Setting Up Your React App with Vite

✅ What You'll Learn

- Create a Vite-powered React app
- Understand the project structure
- Push your code to GitHub

🔍 Why It Matters

Starting cleanly saves time and ensures future deployment is smooth.

□ Before You Begin

- Terminal access and GitHub login ready

👥 Step-by-Step

1. Run:

```
npm create vite@latest my-app-name -- --template react
cd my-app-name
npm install
```

npm run dev

2. Push to GitHub

💭 Reflect + Extend

- What do each of the folders/files in your new Vite project do?

🚀 Real-World Task

Take a screenshot of your new app running and your GitHub repo.

📘 Chapter 3: Moving HTML/CSS into React

✅ What You'll Learn

- Convert HTML into React components
- Import your CSS
- Organize your folders

🔍 Why It Matters

Components make your site modular and scalable.

👥 Step-by-Step

1. Create a `components/` folder
2. Move static HTML sections into components (e.g., Header, Footer)
3. Import and apply CSS

💭 Reflect + Extend

- Which parts of your HTML are reusable?

🚀 Real-World Task

Break down your homepage into at least 3 React components.

📘 Chapter 4: Replacing Vanilla JS with React Hooks

✅ What You'll Learn

- Convert JS interactions to useState/useEffect
- Control form inputs

🔍 Why It Matters

React replaces direct DOM manipulation with a declarative, scalable pattern.

👥 Step-by-Step

1. Convert a click event (e.g., counter) to useState
2. Use useEffect to track changes
3. Convert a contact form into controlled components

💭 Reflect + Extend

- What JS functions did you replace with React logic?

🚀 Real-World Task

Rebuild one JS-based feature using hooks.

📘 Chapter 5: Creating the Backend with Node + Express

✅ What You'll Learn

- Set up a basic Express server
- Handle simple API routes

🔍 Why It Matters

Your backend powers your logic, data, and communication with the frontend.

👥 Step-by-Step

1. Create `/server/index.js`
2. Add a route:

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

3. Test with browser or Postman

💭 Reflect + Extend

- What parts of your site will need backend logic?

🚀 Real-World Task

Build and test a contact POST endpoint.

📘 Chapter 6: Connecting MongoDB Atlas

✅ What You'll Learn

- Connect a cloud database
- Build and use a schema

🔍 Why It Matters

Data lives in MongoDB. This is how you make your site dynamic and persistent.

👥 Step-by-Step

1. Create a MongoDB Atlas account
2. Create a Mongoose schema and connect to the DB
3. Save and retrieve test data

💭 Reflect + Extend

- How would your project change if it could store user input or content?

🚀 Real-World Task

Store a submitted message in your MongoDB collection.

📘 Chapter 7: Full Stack Integration

✅ What You'll Learn

- Fetch data from your Express API in React
- Display and submit data from/to MongoDB

🔍 Why It Matters

This is where it all connects: client ⇒ server ⇒ database and back.

👥 Step-by-Step

1. Use `axios` in React to call your API
2. Render dynamic data (e.g., messages)

3. Submit a form that saves to MongoDB

💭 Reflect + Extend

- How is React now responding to backend data?

🚀 Real-World Task

Build a working feedback form with success/error messages.

📘 Chapter 8: Deployment

✅ What You'll Learn

- Deploy frontend with Vercel
- Deploy backend with Render
- Connect everything with .env securely

🔍 Why It Matters

Launch your app so the world can see it and use it.

👥 Step-by-Step

1. Push frontend to GitHub and deploy with Vercel
2. Deploy backend on Render
3. Add environment variables (e.g., Mongo URI)

💭 Reflect + Extend

- What does deployment expose (errors, configs, speed)?

🚀 Real-World Task

Create and share your live site and repo links.

📘 Chapter 9: Capstone Project — Convert a Static Site

✅ What You'll Learn

- Plan and complete a full-stack project from a static base

🔍 Why It Matters

This is your proof of learning — a real upgrade from static to MERN.

👥 Step-by-Step

1. Pick a static site (e.g., portfolio, blog)
2. Identify static vs. dynamic parts
3. Convert each section:
    - HTML → React
    - JS → Hooks
    - Forms → API
    - Content → MongoDB
4. Deploy both frontend and backend

💭 Reflect + Extend

- What would you do differently next time?

🚀 Real-World Task

Record a Loom demo or write a blog post showcasing your conversion process.