Track Web01

Final Product :

Build a Note Making WebApp, Where users can take notes, save them, edit them later, and also can share them with their friends via the link.

Frontend :

- The front end should consist of a minimal-looking dashboard where all the notes are listed in an alphabetical/chronological manner.
- Note Page: Consist of note title, creation date, last updated date, and the note content.

Backend :

- Create APIs to provide the required data to the frontend part.
- APIs must be protected so that personal notes' content isn't visible to the public.
- Integrate Google authentication and other types of login/sign-in features.

Extras :

- Rich-text formatting / Markdown Editor for the notes.
- Friends are able to comment on the shared notes.
- A good-looking User Interface.

Recommended Tech Stack :
ReactJS, NodeJS, ExpressJS, MongoDB (MERN Stack)

1. **Front End Web Technologies** :

" The best way to learn is by implementing things, so instead of just reading these, do try implementing each of them."

Step 0: Basics
If you've never touched frontend technologies before, you'll have to go learn those before. It's going to take time. But you won't be able to move forward without it. Mozilla Developer Networks is your friend here. ***Stay away from w3schools because it usually just covers the syntax instead of explaining the topics. You should always look for concepts in MDN docs.***

- HTML Basics
- CSS Basics
- Javascript Basics
- HTTP Basics

Step 1: Basics Continued...
Now that you know the basics, you'll continue your journey to learn different day-to-day concepts that you'll need while programming. I encourage you to understand these concepts deeply. You'll always need them as long as you work in the frontend domain.

- HTML / Browser
    - Introduction to the DOM
    - Introduction to events
    - Web Storage API (local storage)
    - History API

- CSS
  This is a topic that is always easy to start with but difficult to have a good understanding of. While working with CSS, there are often hacks/forceful styling which seems to work well for us but for building a good application, we should use CSS in such a way that works well with all the different screen sizes. The given articles are a must-read and will provide you a good understanding of important concepts.
  - How to learn CSS
  - Specificity explained
  - A guide to flexbox

- Javascript
  - Book series - You Don't Know JS
  - Book - Eloquent Javascript
  - Understanding hoisting in Javascript
  - What is function composition
  - Closures and scoping
    - Additional reading: How do js closures work? - StackOverflow
  - Map, reduce and filter

- Networking
  - Client-Server overview
  - XMLHttpRequest
  - Working with JSON

Step 2: Specific Interesting topics

This is a collection of some interesting concepts that are often used. You must know what's an event loop in an asynchronous environment. Similarly, module systems, routing, authentications are all interesting and important concepts. This list is nowhere near exhaustive. But it'll give you a good understanding of where you are. Frontend development has evolved so much. Just learning Javascript, HTML, CSS doesn't cut it anymore.

- What the heck is the event loop anyway?
- Javascript Module systems
- Building a basic javascript router.
- Networking
    - What are web sockets?
    - Session VS token-based authentication

Step 3: The next layer

The topics in this section are about the advance tech which gets mostly used while building web applications. Before proceeding into these, you should have a good understanding of why we need to use them in the first place - their advantages and disadvantages if any. There are many different frameworks that are being used, you can check them all and decide which one suits you best and start with. The following links are concerned around - react, redux and sass.

- Tutorial: Intro to react
- React: getting started
- React lifecycle methods
- React router

    - You can also check for anything related to the router: React training: react-router
- SASS basics
- Redux

You should have a clear understanding of how data in these frameworks flows between components and why is it done in such a manner.

Step 4: Behind the scene

Frontend space is vast and there is a lot to consume but it is always important to have an idea of what happens behind the scenes and the understanding of how the browser works.

- What happens when the g-key is pressed?
- How do browsers work?
- Critical rendering path
- The secrets of CSSOM and why you should care!
- Javascript engine: how do they even?

2. **Back End Web Technologies** :

    - The first thing to start with would be to have a good overview of how frontend and backend work together. What exactly happens in a client-server architecture when you hit a URL and how is the response served from the backend.
    - One part of being a good developer is to have a good idea of how to use the browser's developer tools. So do

try your hands on it. Because they are a great help while debugging.

- Once you are good with the understanding part of it you can start exploring the backend tech. The most basic way to start with is by learning to build applications that can perform CRUD operation - Create, read, update and delete.
- Understand different types of requests - GET, POST, PUT, etc and build a strong understanding of what to use when.
- Integrating backend and frontend is an important part while building web applications so make sure to have an idea of it while trying to implement basic applications.

a. **Easy**: Firebase
b. **Medium**: Flask (Python-based)
c. **Advanced**: NodeJS, Django
   NodeJS: Starting with nodejs, you must have a good understanding of JS and its concepts. Such as - Callback, Promises, async/await, closures, etc.
   While learning NodeJS, first get to know why NodeJS, how is it built, how is it better than other Backend(Runtime Environment) Frameworks.
   Then after you have a basic understanding of working with nodejs, you can explore npm modules and how to use them.

   Django:
   It would be easier if you practiced Flask services, most would be the same, except Django is more scalable and more robust and provides a bit more flexibility and direct DB connections.

The best way to learn Django is by implementing projects, be it as simple as To-Do application, etc. Check youtube channels like for more Django content: CS Dojo, Dennis Ivy, Codemy

Good resources you can look up for web development:

- https://www.freecodecamp.org/learn
- Florin pop: https://www.youtube.com/channel/UCeU-1X402kT-JlLdAitxSMA/
- Akshay Saini: https://www.youtube.com/channel/UC3N9i_KvKZYP4F84FPIzgPQ/videos
- TechSmith: https://www.youtube.com/channel/UCbGZKLIHpox2l0whz6_RYyg