Welcome Folks!

Following are the topics we are going to cover in this lesson

1. What is Full Stack Web Development?
2. Difference between Software Developers and web developers
3. Career in Full Stack Web Development.
4. MERN stack Popularity & importance
5. What should you learn and in what sequence?
6. How to get better at learning software?
7. Tools required for upcoming sessions.

What is Full Stack Web Development?

Web developers are primarily responsible for designing and developing web applications and websites, right from their user interface, functionality, back-end working, to even how it gathers and processes data. Every company that has to put itself on the map today needs an online presence and what better way to do so than having a strong presence on the internet via an elegant website?

Following are some of the common web developer job responsibilities:

- Developing new web design projects by coordinating with clients and business leaders
- Designing and testing the functionality of web applications and websites with the use of web development tools.
- Ability to write code in HTML, CSS, JavaScript, and other web development programming languages.
- Providing easy methodologies to integrate a variety of content such as images, audio, and video onto the website
- Providing technical support for website deployment
- Coordinating with experts to provide top-notch security elements to protect business assets and website data.
- Making use of a variety of tools to monitor website traffic, insights, and performance.

The web development field is split into three specialized areas based on professional interests, skill sets, and areas of expertise:

1. Front-end web development
2. Back-end web development
3. Full-stack web development

Front-end web development deals with designing the look, the feel of the website, and the web applications from the users' perspective. Hence, all of the things are considered such as the access to content, ease of navigation and elegant functionality of elements on the website, and more.

Back-end web development deals with the technicalities of the website that help in providing the functionality. Database handling, creation of application logic, integrating and creating new APIs based on specifications, and development of the overall website architecture are what back-end development is all about.

Full-stack web development, as the title suggests is a career path that involves proficiency in both front-end and back-end development tasks. Extensive experience is required in the field of user experience, programming, and application design in general.

It is very vital that you clearly understand the difference between software developers and web developers.

What is Software Development?

Software developers are responsible for inventing, managing, and optimising applications that run on various devices such as desktops, smartphones, and other mobile devices.

The design and development of software are driven by the usage of a variety of programming languages such as Python, C#, Java, and SQL. Code is sometimes complex as it has to govern the entirety of the end-to-end functionality of the software.

Many software technology companies like Oracle and Microsoft have designed products that help in building user interfaces, creating applications, troubleshooting bugs and errors, and even automating many steps in the process of designing.

Following are some of the common software developer job responsibilities:

- Designing software-based solutions as per users' needs.
- Collaborating with experts to recommend software updates to existing systems
- End-to-end documentation of new applications and system design
- Creating detailed models that outline the programs and their functionality
- Ability to effectively test programs for consistent functionality and efficiency
- Ability to merge existing systems with the new software

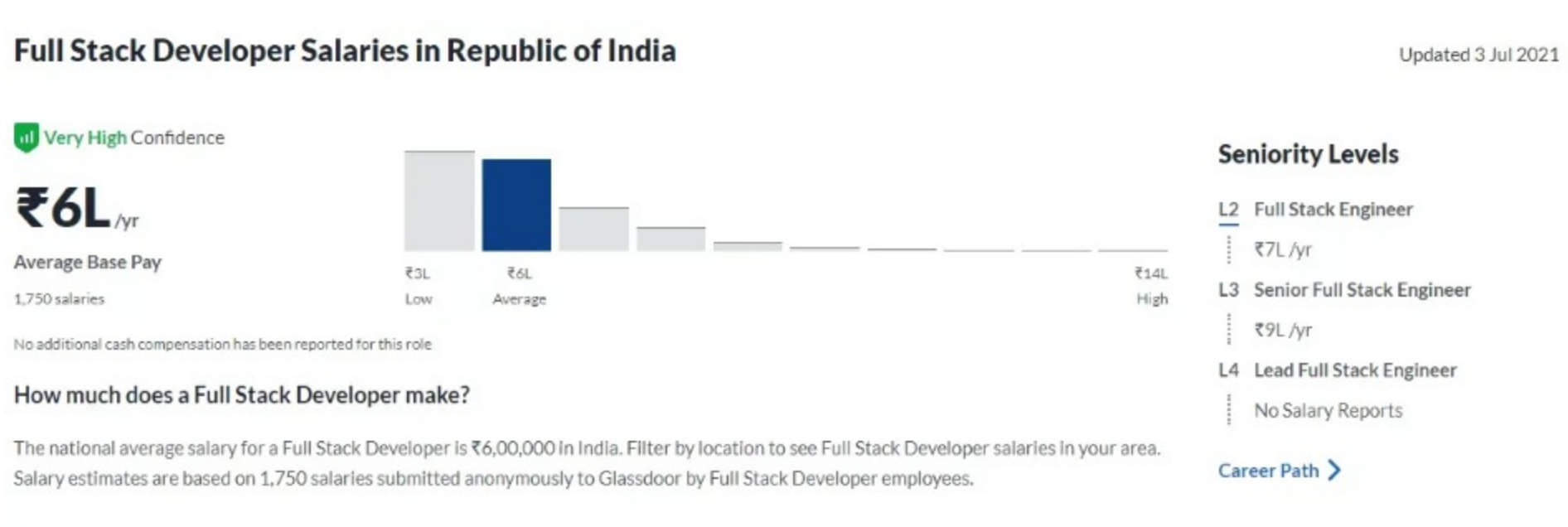
Next up, let us understand the difference between a software developer and a web developer considering a variety of points.

Web Developer vs Software Developer

As discussed in the above sections, both of these roles are quite different to each other in many factors.

Key Points	Web Development	Software Development
Ability	Web Developers' skills include the ability to develop websites and web applications only.	Software Developers can work around and build any kind of software that is needed by the organization or client.
Architecture	Web Developers are known for creating and working with the development of client-server architectures.	Software Developers mostly focus on providing solutions to clients by working on client-based systems.
Development Platform	Web Developers work with the mindset to develop applications that work flawlessly across a spectrum of web browsers.	Software Developers ensure that their products and applications have the ability to work well on different OS platforms.
Easy to Learn	Web Development is generally easier to get started with as the tools and techniques are not complicated to learn.	Software Development requires an in-depth understanding of tools, methodologies, and philosophies that govern good software
Programming Languages	HTML, CSS, JavaScript, React, Node, and other web development tools and frameworks	C#, C++, Java, Python, C#, Golang, and more

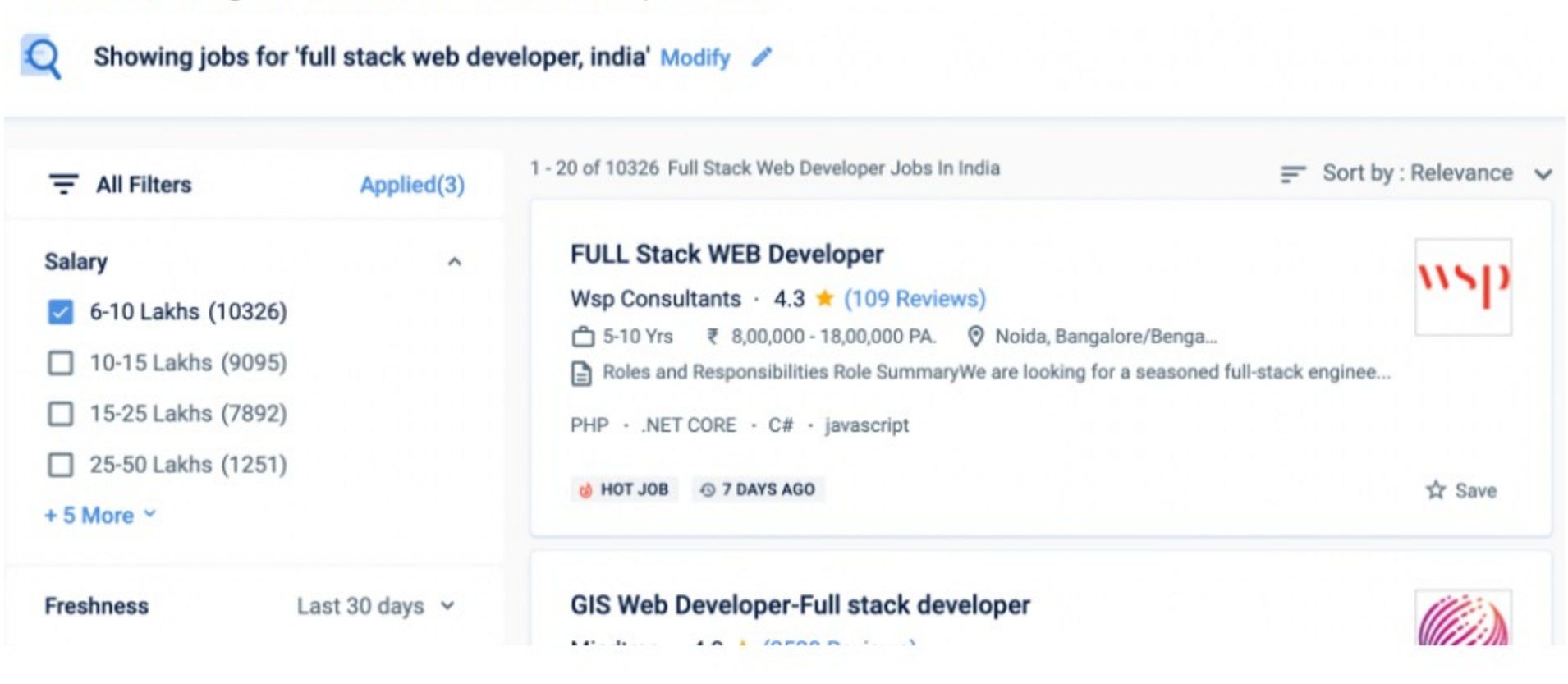
Career in Full Stack Web Development



- As a Fresher entry-level | 0-1 year | Full-Stack developers | the average salary in India is INR 5,75,000 – INR 6,00,000
- A full-stack developer with 1-4 years of experience, earns INR 650,000 – INR 15,00,000
- An employee with 5-9 years of experience can earn 15 – 30 Lakhs

Here's some more information on how much top companies pay for full-stack development employees:

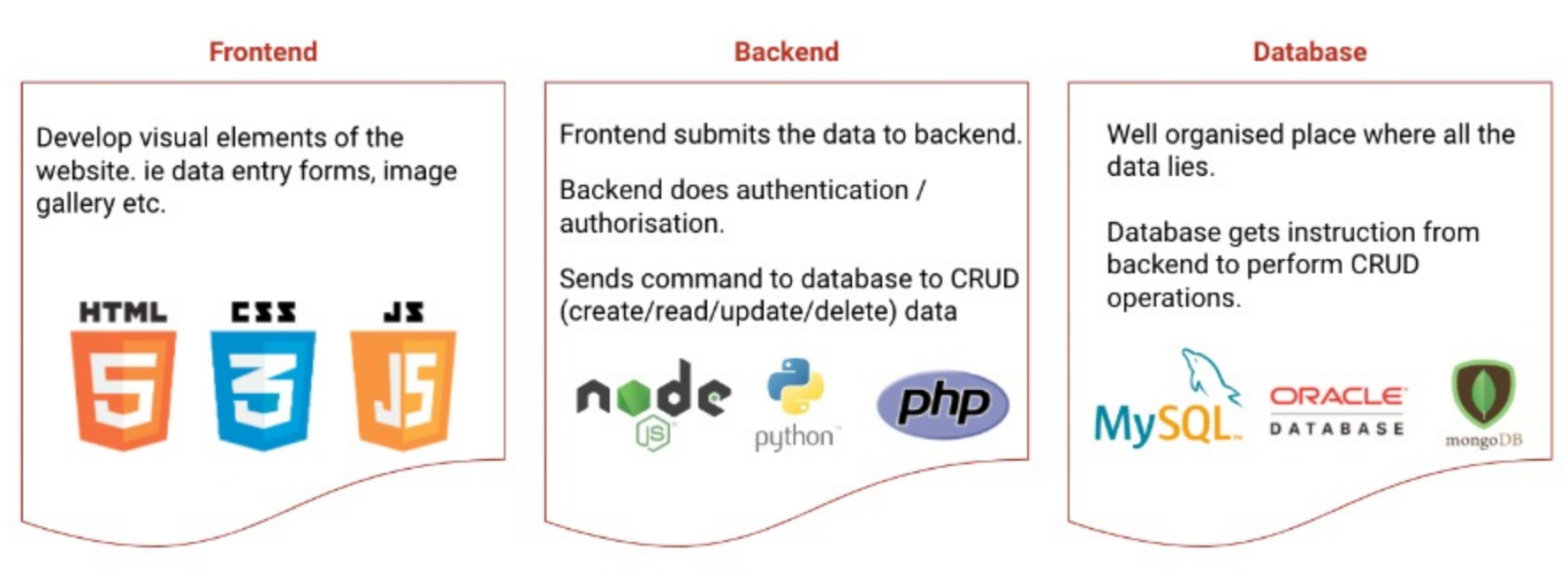
- Tata Consultancy Services - INR 4,52,846 per annum
- Vassar Labs- INR 7,85,542 per annum
- Nuclei- INR 10,55,280 per annum
- IBM – INR 6,59,371 per annum
- Report Garden- INR 8,09,013 per annum
- Accenture- INR 5,30,241 per annum
- Wipro- INR 4,20,826 per annum
- GALE Partners- INR 81,8,863 per annum
- Cognizant Technology Solutions-INR 2,40,000 – INR 6,25,000 per annum
- Oracle – INR 13,90,000- INR 16,93,000 per annum
- Schlumberger- INR 16,80,000- INR 21,17,000 per annum
- Infosys – INR 3,35,000 – INR 4,24,000 per annum
- Tapzo- INR 11,00,000- INR 12,10,000 per annum
- Posist Technologies- INR 8,34,000- INR 9,95,000 per annum



In the last 30 days, more than ten thousand full-stack web development jobs have been listed on Naukri.com. There is ample demand for everyone to pursue this field.

Mandatory Skills you need as a Full Stack Web Developer

For any web application, development work is divided into 3 parts. Frontend, Backend & Database.



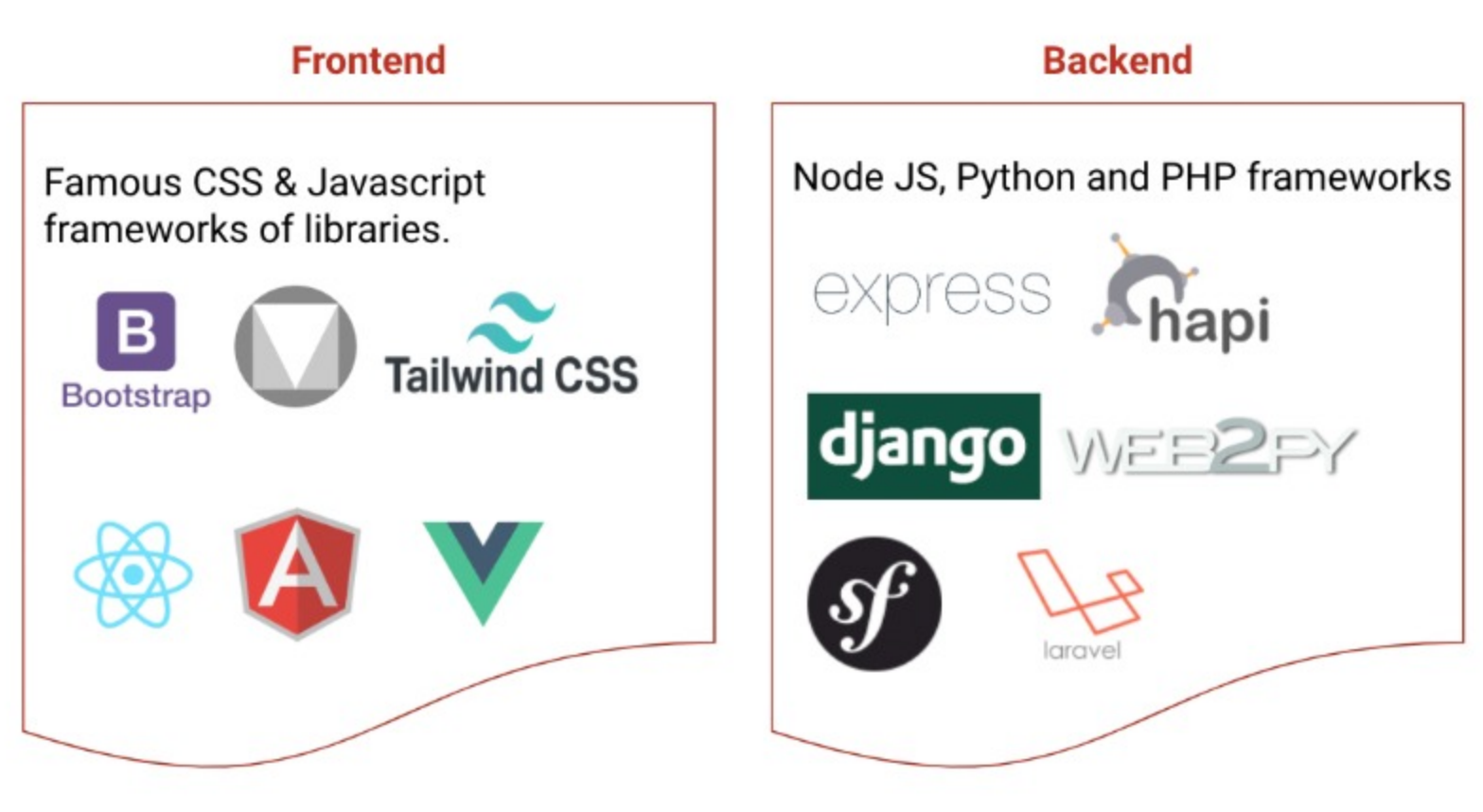
Frontend: HTML, CSS, Javascript

Backend: Node.js, Python, PHP

Database: MySQL, Oracle, MongoDB

Full-stack engineers use frameworks for frontend and backend developments.

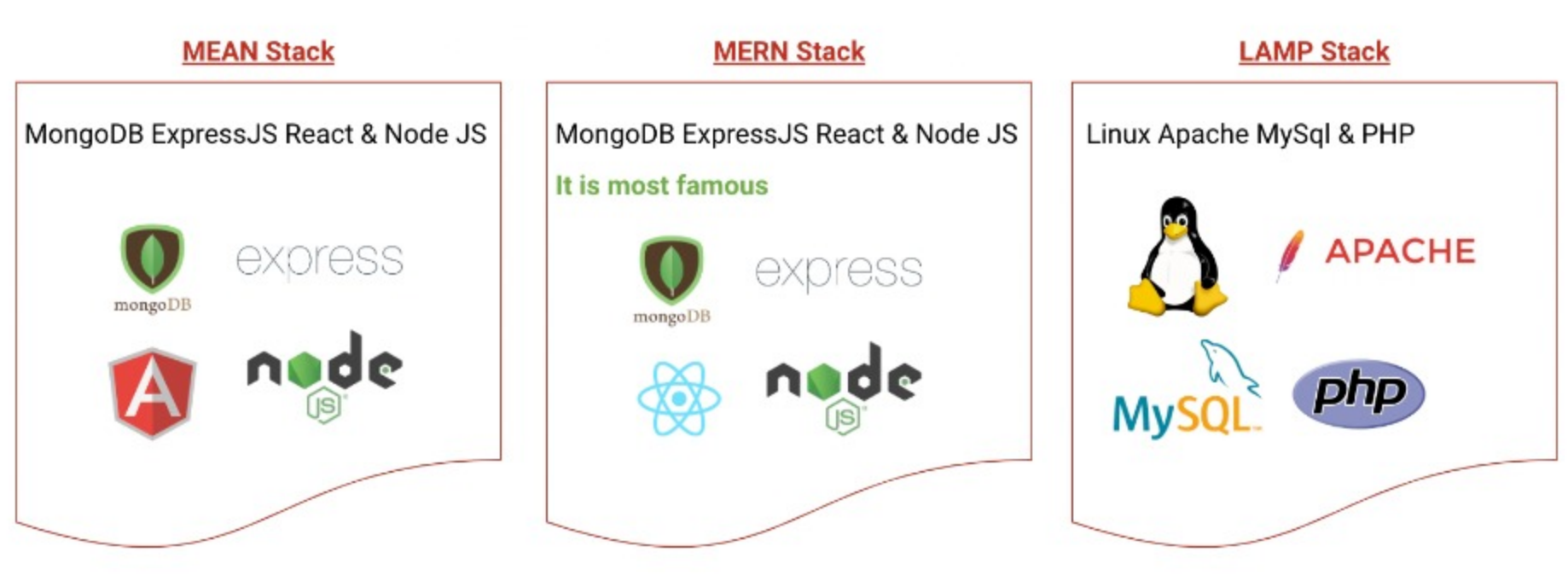
A framework is a ready-made tool that speeds up the development work, makes code well organised & also solves many common problems on behalf of developers.



Frontend: React.js, Next.js, Angular.js, Bootstrap, Tailwind CSS, Vue.js

Backend: Express.js, Django, Hapi, Symfony

Full Stack Engineers use well-known combinations of technologies known as **Stacks** to develop websites.



Why MERN stack is popular among Full Stack Engineers?

MERN stack advantages

- JS is used to span the full development cycle in the MERN stack, from frontend to backend.
- For a smooth **MERN full stack development** process, it supports MVC(Model-View-Controller) architecture.
- In **MERN vs MEAN**, MERN permits developers can develop using only JS and JSON.
- It has a large number of testing tools.
- It has a strong open-source community behind it.

What should you learn and in what sequence?

AlmaBetter has designed the course in such a manner that within 24 weeks students will learn end to end web development. The course has been divided into 5 modules.

Module 1: Programming web with Javascript

In this module, we are going to learn about the fundamentals of web, browser, JavaScript and the basics of data structures and algorithms.

Module 2: Frontend Development

This module will cover HTML, CSS, JavaScript on the Browser and React.js.

Module 3: Backend Development

This module will cover SQL, Node.js, Express.js, Microservices and MongoDB.

Module 4: System Design

This module will cover high-level, low-level scalable design, OS and Networking Fundamentals, Cloud Services, Advanced DSA, and Graph Theory.

Module 5: Blockchain Technology

This module will cover Blockchain Technology, Ethereum, Smart Contracts, Introduction to Solidity, Building React Dapps, Blockchain Cryptography, Ethereum Virtual Machine (EVM), Ethereum Tokens, ERC20 and ERC721 (NFT), Decentralized Finance (Defi).

How to get better at learning programming languages?

1. Make Your Fundamentals Clear

A common mistake that a student or beginner commits while learning programming is skipping the fundamentals or chapter 1 and directly jumping to the next chapter right away. To understand the advanced concepts of programming you need to be very clear about the fundamentals of programming. If you will be doing the same mistake then at some point, you will end up with lots of confusion and you will have to come back to your basics again. These fundamentals are Data Structures, variables, control structures, syntax, tools, or text editors. When you start doing programming pick one programming language, stick with it, and clear all the basics of programming first before going to the next level. Your overall time to learn to code will be definitely saved if you will follow this path. As we will follow the MERN stack, students must pick Javascript as the language.

2. Learn By Doing, Practicing and Not Just Reading

A common mistake beginners do while learning programming is just reading a book or looking at the sample code on their desktop without practising it. It's easy to read about the loops, and variables, and get all the things in your head but actual programming doesn't work in this way. You really need to get your hands dirty in coding and keep practising it regularly. When you start programming you face a lot of problems, you get stuck there, you will be asked to implement the code practically and find the solution for a specific problem and there you will scratch your head while implementing the code. When you write the code, play with the code, change your code to see different results, optimise the code and try different solutions, your logical thinking ability get improves day by day and you eventually learn a lot of things that make you a better programmer. When you start coding, practice the same code or sample again and again until or unless you don't need to refer to the same book or resource from where you have learned.

3. Dry Code by hand

When you start programming as a beginner you will be thinking that why should I code by hand. It's a time-consuming process, I can't run and check my code on paper and also if I actually need to implement something on my system then why should I use pen and paper. Coding by hand is something old-school technique but it actually involves a test for a programmer's proficiency. Coding by hand can give you a clear understanding of syntax and algorithms, you make a deeper connection in your brain. Learning programming this way will make your work easier and faster later.

4. Problem-solving attitude

Problem-solving involves diagnosing the possible causes of a problem and developing an action plan that solves that problem. People use problem-solving skills all the time, both in their personal and professional lives. Effective problem-solving in the workplace often requires following a step-by-step process and using a designated problem-solving framework.

- ***Focus on the solution:*** It's easy to become hyperfocused on the conditions that created the problem. Shifting your focus away from the current problem to possible outcomes and solutions can give you a more positive outlook and open your eyes to new solutions.
- ***Clearly define the problem:*** It's hard to solve a nebulous problem you never took the time to clearly define. No workplace is perfect, and there are usually a variety of interrelated problems that can be solved at any one time. If you find yourself getting overwhelmed and distracted during the problem-solving process, go back to step one and make sure you are approaching a singular problem.
- ***Agree on a process:*** If you're problem-solving as part of a team, it's very important that you agree to basic ground rules and procedures before you start the problem-solving process. This will streamline the process and help you prevent conflict down the road.
- ***Take Breaks:*** If you are solving problems it's not good to sit in front of a computer for hours and hours and try to grasp everything in one go. You will be exhausted by doing this so it's better to learn coding in chunks. Take some short breaks to get refreshed. This will help you look at the problem from a different angle.

5. Following best practices

Students should browse good open-source projects on GitHub to follow the best practices. The most important aspect of any project is code readability. It is key to maintainability and working together with a team. Open-source projects are built with the input of many developers. These projects need to maintain a high level of code readability so that the team can work together as efficiently as possible. Students can follow this project on Github: Ember.
Few best practices followed by industry professionals are:

- Commenting & Documentation
- Consistent Indentation
- Code Grouping
- Consistent Naming Scheme
- Don't Repeat Yourself Principle
- Avoid Deep Nesting
- Limit Line Horizontal Length
- File and Folder Organisation
- Consistent Temporary Names
- Use Meaningful Names for Variables and Functions
- Avoid hardcoding values

6. Time management

Time management is the ability to use your time productively and efficiently. You could also think of it as the art of having time to do everything that you need, without feeling stressed about it. It sounds simple, but it is much harder in practice.

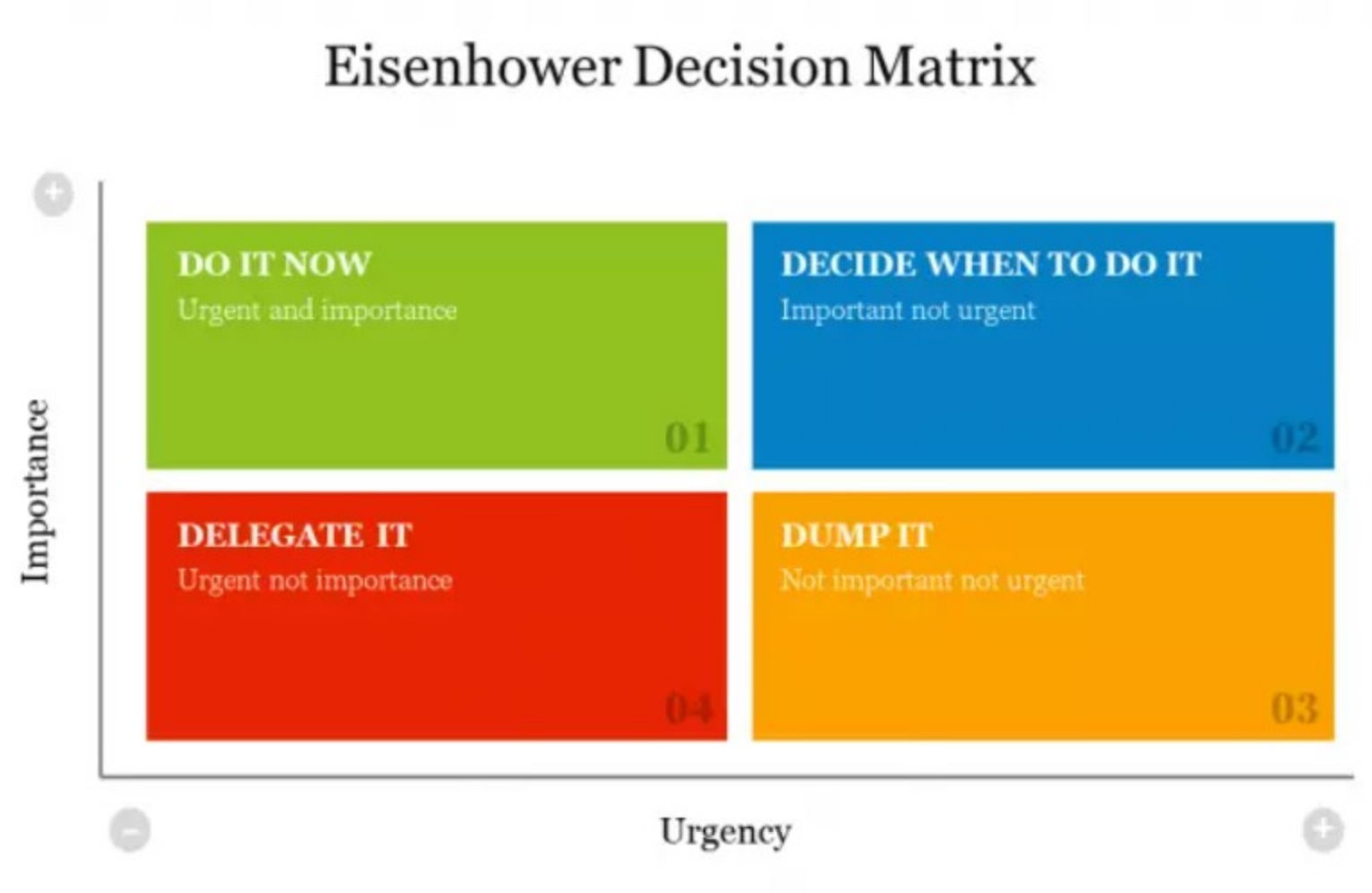
Time management skills are essential because few, if any, of us ever have enough time to do everything that is asked of us, or that we want to do.

Time management is defined as using your time productively and efficiently—but what about when you are working as productively as possible, and you *still* can't get everything done? It may be better to think about time management as a combination of working productively and prioritising your time.

The five most important time management skills are:

- **Planning**: The best way to develop planning skills is to consistently use a calendar tool, such as Google calendar. Simply plan your week ahead, directly in the calendar: When to wake up. Which task to start first in a day. When to take meetings. Your leisure time. And so on.

- **Decision making and prioritization:** The Eisenhower matrix is probably the best start for learning prioritization. The matrix recommends arranging tasks in one of the four quadrants:



- Urgent + Important (Do) - Urgent + Not Important(Delegate) - Not Urgent + Important (Decide when to do it & Put it in your calendar) - Not Important + Not Urgent(Dump it)

List all the tasks you have to perform in a week. Arrange them in one of the four quadrants. In every quadrant, sort them from the most important one to the least important one. Start with the most important and urgent tasks. That's how you'll practice making decisions about your time.

- **Setting boundaries and saying no:**

That means setting very clear boundaries for people who want to delegate tasks to you or engage you in projects and activities that are not your priority. In this regard, saying no is one of the most important skills in time management. The best way to develop good boundary-setting skills is to start saying no in small unimportant situations, and then slowly scale up to the bigger things where you are more emotionally involved.

- **Delegating and outsourcing tasks:**

Saying "no" is about protecting your time. Delegating tasks is about leveraging other's people time. Learning how to delegate is a very important next step in being a master of time management.

The main point of task delegation is that you can focus on the highest value activities and get all the rest off your back. It's a kind of specialization.

With proper delegation, you can also engage people who are bigger experts than you are at some things, meaning they can perform some tasks better and faster.

- **Building a system and diligently following it:**

Research all the time management techniques out there. Select the ones you find the most interesting. Start testing them one by one in practice and give yourself a few months to select the techniques that fit you best.

Absolutely combine time management techniques with different tools that support the selected techniques; that can be different calendars, note-taking, time management apps and other apps in combination with reminders, notifications and checklists. Whatever suits you finalise and follow it diligently.

7. Sharing your work & Building your network

One of the best ways to understand programming easily and quickly is by sharing your work with others and getting feedback. Students can utilise AlmaBetter's community to connect with fellow coders and learn together. Teaching someone, sharing your knowledge, and doing discussions with other programmers will make you a better programmer quickly. Teaching to someone is teaching to yourself too, so if you are able to teach to somebody that means you truly understand the concepts. It is the best habit to learn something in-depth and you will realise that you don't need to come back on the same topic.

You can also participate in open source projects, discuss your code with your co-programmers or contribute on Github, also you can take help from forums or discussion sites. When you learn programming do not hesitate in asking for help from the community and make a habit of reading programming related blogs and taking help from Stackoverflow, Reddit's "learn programming" subreddit. Beginners do this mistake and feel shy when they need to ask for help. It doesn't matter if you ask silly questions and look stupid, it will help you in the long run and if you don't do it you will be struggling in coding later. Connect with mentors or take help from fellow programmers to understand concepts easily and quickly. Your mentor or professional can guide you better because they have been already in your shoes before.

Tools required for upcoming sessions

1.**VS Code:** IDE for coding, Download and install before next session. Click Here

2.**Github:** Online software development platform used for storing, tracking, and collaborating on software projects. This platform will be used for Capstone project submission.Sign up and explore before next session. Click Here