1. Base Languages and Standards

Front-end work starts with three base languages: HTML, CSS, and JavaScript. HTML is used for structure, CSS for styling, and JavaScript for behavior. Most projects today use HTML5 and CSS3 standards. HTML5 defines a set of elements and attributes that are processed by browsers, and CSS3 provides new layout options such as flexbox and grid. JavaScript is implemented in the ECMAScript 6 (ES6) standard and later versions, and these standards add support for classes, arrow functions, promises, and modules. Many large websites use HTML files that are around 50–150 kilobytes in size once they are fully processed by the browser.

2. Core Frameworks and Libraries

Modern front-end development makes use of frameworks and libraries. The following are examples with specific details:

React:

React version 18.2.0 is common in many projects. The minified and gzipped core library is roughly 35 kilobytes. React uses a component-based structure that breaks the user interface into parts that can be individually updated. Its virtual DOM diffing algorithm processes changes in about 1–3 milliseconds for moderately sized components. In real-world applications, state updates in React may run in under 20 milliseconds per operation in most cases.

Angular:

Angular, with version 14.x in use in some projects, compiles TypeScript into JavaScript and produces bundles that can be around

200 kilobytes after tree shaking and compression. Its change detection process may inspect hundreds of elements per cycle and typically completes a cycle in 16 milliseconds in small to medium applications. Angular also enforces strict typing and module boundaries, leading to code bases that can range from 50,000 to over 500,000 lines of code in enterprise systems.

Vue:

Vue 3.x is built using a virtual DOM approach similar to React. Its core library is near 25 kilobytes after minification and compression. Vue allows single file components where template, script, and style can be defined together. The reactivity system in Vue operates with updates processed within 5–10 milliseconds per change when a component is updated.

Other libraries and micro-frameworks are in use on projects where minimal overhead is desired. Some projects use Svelte, which compiles components to plain JavaScript. Svelte-generated code might be 20–30 kilobytes for basic functionality and avoids the runtime overhead typical in other frameworks.

3. Build Tools and Bundlers

Modern front-end projects rely on build tools that process code and assets. The following are some commonly used tools:

Webpack:

Webpack is configured to bundle multiple JavaScript files and dependencies. Large code bases may result in bundle sizes of 200–500 kilobytes uncompressed. When tree shaking and

compression are applied, the final bundle might be in the range of 50–150 kilobytes. Build times with Webpack can vary: a mid-sized project may require 30–60 seconds on a machine with a quad-core CPU and 16 GB of memory.

Vite:

Vite is used as an alternative with faster startup times. In many cases, Vite completes module processing and hot module replacement (HMR) in 1–3 seconds on development servers for medium-sized applications. The final bundle sizes remain similar to those produced by Webpack after production build steps.

Parcel:

Parcel automatically detects dependencies and performs parallel processing. It is used on projects where configuration overhead is minimized. Build times with Parcel for a project with 100+ modules typically fall in the range of 20–40 seconds.

These tools perform code transpilation, asset compression, and module concatenation. They use plugins such as Babel to convert modern JavaScript syntax into a form that is compatible with older browsers. Babel often runs in under 100 milliseconds per file on a modern processor when processing code with around 1,000 lines.

4. CSS Processors and Preprocessors

Styling can be managed with plain CSS or with preprocessors that offer additional features:

Sass (SCSS):

Sass files with .scss extension are compiled into CSS. In small projects, a single .scss file of about 2,000 lines may compile in less than 500 milliseconds. Larger projects, with up to 50,000 lines spread over multiple files, might take 2–3 seconds to compile.

Less:

Less compiles similarly to Sass and is used in many legacy systems. Its compilation times are comparable to Sass.

PostCSS:

PostCSS is used to process CSS for autoprefixing and other transformations. On average, processing a file of 100 kilobytes takes less than 200 milliseconds on a typical development machine.

- PostCSS API

5. Testing and Debugging Tools

Quality assurance in front-end development requires tools for unit testing, integration testing, and end-to-end testing:

Jest:

Jest is commonly used for unit testing JavaScript. On a machine with 8 cores, running a test suite with 500 test cases can complete in 10–20 seconds. Coverage reports generated by Jest often aim for a threshold of 80–90% line coverage.

- Jest Testing Framework

Cypress:

Cypress is employed for end-to-end testing of user interfaces. A typical test run of 50 tests might take 30–60 seconds, depending on the number of interactions simulated.

- Cypress End-to-End Testing

ESLint and Prettier:

These tools check code quality and format style. ESLint can process files with around 1,000 lines in less than 50 milliseconds per file. Prettier formats files in about 20–30 milliseconds per file on average.

- ESLint Rules
- Prettier Configuration

6. State Management and Data Handling

In front-end applications, data handling often requires state management libraries:

Redux:

Redux manages state using a single store that can be as small as 10 kilobytes in memory or grow to 100 kilobytes in larger applications. Middleware in Redux can add 5–10% overhead to processing time per action, and state updates typically complete in less than 5 milliseconds.

- Redux Toolkit

MobX:

MobX uses observable data and reaction-based updates. In a project with 1,000 observable properties, updates may be processed in 3–6 milliseconds per change event.

- MobX Guide

Other Tools:

Some projects use tools like Zustand, which offer a simpler state management system with a footprint of 5–8 kilobytes in minified code.

- Zustand

7. Code Optimization and Performance

Developers use techniques to reduce bundle sizes and improve runtime performance:

Tree Shaking:

Tree shaking removes unused code. For example, a library that starts at 150 kilobytes may reduce to 40 kilobytes if only 25–30% of its functionality is used.

- Tree Shaking in JavaScript

Lazy Loading and Code Splitting:

Breaking an application into smaller parts that load on demand can reduce initial load times. An application that initially loads 200 kilobytes might delay loading additional modules until user interaction, keeping initial load times under 1 second on a broadband connection.

- React Code Splitting

Compression:

Gzip or Brotli compression reduces bundle sizes by a factor of 3–4. A file that is 120 kilobytes uncompressed might be 30–40 kilobytes when served with Brotli.

- Gzip Compression
- Brotli Compression

Caching:

Static assets are often cached on the client side. Using service workers, files with sizes in the range of 50–150 kilobytes are cached locally to reduce repeated downloads. Cache expiry values of 7–30 days are common in many systems.

- Service Workers

8. Developer Tools and Local Development

Development environments use a range of tools to improve coding efficiency and debugging:

Browser Developer Tools:

Tools built into Chrome, Firefox, or Edge allow inspection of network requests, JavaScript performance, and CSS layout details. In some cases, developers observe JavaScript execution times of under 2 milliseconds per function call using these tools.

- Chrome DevTools
- Firefox Developer Tools

Source Maps:

Source maps allow the browser to map minified code back to original source files. A project with 100 source files may produce a source map file of 1–2 megabytes in total size, which is usually served only in development mode.

- Source Maps Explained

Hot Module Replacement (HMR):

HMR systems refresh only changed modules. In a project with 200 modules, an update may take 100–300 milliseconds to compile and apply, keeping the workflow efficient.

- HMR in Webpack

9. Continuous Integration and Deployment

Automation in testing and deployment forms part of the tech stack:

CI Tools:

Tools such as Jenkins, Travis CI, or GitHub Actions run tests and builds on code commits. A build pipeline for a mid-sized project might run in 3–5 minutes and produce detailed reports on test coverage, bundling sizes, and error rates. Pipelines often run 50–100 tests in a single job.

- GitHub Actions
- Jenkins
- Travis CI

Deployment Platforms:

Deployment services like Netlify, Vercel, or AWS Amplify can push updates in 30–60 seconds after a build completes. These systems support rollbacks and can handle traffic peaks with backend scaling. A typical deployment might serve files with sizes between 10 kilobytes and 2 megabytes depending on the module.

- Netlify
- Vercel
- AWS Amplify

10. Package Management and Version Control

Managing dependencies and source code is essential:

Package Managers:

npm and Yarn are used to install and update libraries. In a project with 200–400 dependencies, the installation process can take 30–90 seconds on a standard broadband connection. The lock file generated

(package-lock.json or yarn.lock) may be 100–500 kilobytes in size, ensuring reproducible builds.

- npm Documentation
- Yarn Guide

Version Control:

Git is the tool of choice. A repository for a large front-end project might consist of 20,000+ commits, 10–50 branches, and a total size of 100–500 megabytes. Code reviews and merge requests are standard parts of the workflow.

- Git Documentation
- GitHub Guide
-

11. Performance Metrics and Measurements

Developers measure performance using precise numeric values and benchmarks:

Bundle Sizes:

React bundles typically have a gzipped size of around 35 kilobytes, Angular bundles are closer to 200 kilobytes, and Vue bundles are about 25 kilobytes in similar conditions. These values change with the addition of custom components and libraries.

Build Times:

A project built with Vite may complete in 3–5 seconds for incremental changes, whereas Webpack might require 30–60 seconds for a full rebuild in large applications. Developers record these times to monitor improvements.

Runtime Performance:

Front-end performance is measured using metrics such as Time to First Byte (TTFB), First Contentful Paint (FCP), and Largest Contentful Paint (LCP). Optimized pages can record FCP values as low as 500 milliseconds and LCP under 1.5 seconds on broadband connections. Interactions in well-tuned applications often process in 2–5 milliseconds per update.

- Google Lighthouse
- Web Vitals

Memory Usage:

During development, browsers typically allocate 100–300 megabytes for complex applications, while production builds run in environments with 150–400 megabytes of memory usage in the worst-case scenarios.

- Chrome Performance API

12. Practical Examples and Data

Projects often use detailed numeric tracking for each element of the tech stack. For instance, a project using React might log:

- 35 KB for the React core library (minified and compressed)
- 120 KB for additional libraries (such as Redux, React Router)
- 150 KB for combined CSS assets after processing
- A build pipeline that takes 45 seconds for a complete production build A project using Angular might record:
- 200 KB for the main bundle after AOT (Ahead-of-Time) compilation
- 50 ms for each change detection cycle on average
- 60 seconds build times on a typical CI pipeline These measurements are used to benchmark progress and detect

regressions. Developers use browser performance APIs and network monitors to collect data at runtime, with thresholds set for each metric. Tools that measure memory consumption and CPU usage are integrated into development dashboards, and alerts are set if any measured value exceeds the predetermined limit (for example, a bundle size increase of more than 10% from the previous build).

Here are tools to measure and optimize these aspects:

- Bundlephobia (Check Bundle Size)
- Web.dev Performance Score
- Chrome User Experience Report

13. Summary

This article has covered the front-end tech stack in detail. The discussion begins with core languages (HTML, CSS, JavaScript) and standards (HTML5, CSS3, ES6), moves through specific frameworks like React, Angular, and Vue, and then explains the role of build tools such as Webpack, Vite, and Parcel. Testing tools (Jest, Cypress, ESLint) are described along with state management libraries (Redux, MobX) and performance optimizations (tree shaking, lazy loading, compression). Each component includes concrete numeric details: file sizes measured in kilobytes, build times in seconds, and processing times in milliseconds. Package management with npm/Yarn and version control with Git are explained with specific repository sizes and installation times. The article ends with performance metrics that developers track to maintain efficient code. This detailed overview is intended to provide a practical view of the front-end tech stack with data and examples that can be applied in real projects.

The choice of your tech stack determines the sort of product you can develop, how effectively it can operate, and ultimately determines its viability and prosperity. Let's try to understand in more detail why selecting the right technology stack is so important. We've highlighted several essential reasons for you:

> Choice of technology stack determines the product's performance, security, and software reliability.
> The technologies chosen as part of a future product's core tech stack can significantly impact that product's ability to scale effortlessly.
> If you eventually change your mind regarding the chosen tools, it may be rather challenging to change the technology stack that you've already picked. For example, it is not a simple operation to convert a whole system from Ruby to PHP. Thus, choosing the right technology stack based on your product's needs and requirements can avoid reworks and significantly save time and, therefore, money. Also, we recommend reading our guide about how to outsource PHP web development.
> However, in certain cases, migration from one tech stack to the other is still highly necessary, as the stack chosen at one point in time may be outdated and no longer efficient at the current moment. A bright example of such as scenario is a growing tendency of migrating mobile apps from Objective-C to Swift.

As we can see, all of these factors are significant and demonstrate that a correctly chosen tech stack can bring a significant benefit to your product.

That’s why the choice of tech stack should be approached rather carefully, especially if you are not a tech-savvy product owner. Continue reading to understand the structure of a tech stack and how to decide what technology stack is right for your project.

A tech stack is a set of software tools used by software engineers to build mobile, desktop, or web applications. The technological stack consists of two main parts: front-end and back-end; in simple words, client side and server side of the product. Each part consists of programming languages, libraries and frameworks, and other tech tools. Other components that may compose the structure of the tech stack can be middleware, various third-party integrations, such as BI tools, analytics software, payment gateways, and many others, depending on the needs of the product. Let’s look at the main components more precisely.

Front-end technology stack

All users interact with the client-side of the product, called front-end. This is a visible part of your solution. Examples of front-end technologies:

HTML — a hypertext markup language used to create and display electronic documents.

CSS — it is in charge of the web and single page's format and layout. Font styles, sizes, design, color, and other web page elements are all included.

JavaScript — used to make web pages more interactive. It is a programming language that let's construct dynamic elements on web pages using libraries and frameworks such as jQuery, React, Angular, and Vue.js.

Back-end technology stack

The back-end stack is an invisible part of the product that comprises technologies that operate in the background to store and handle data, process and respond to the incoming requests, and allow the front-end to accomplish its job. Back-end includes the following technologies:

Programming languages: C #, Java, PHP, JavaScript, Python, etc.

Frameworks and libraries: Ruby on Rails, Spring Boot, .NET, Django, etc.

Databases: Microsoft SQL Server, MongoDB, MySQL, PostgreSQL, Oracle, Neo4j, etc.

Web servers: Apache, Nginx, Microsoft's Internet Information Server (IIS), etc.

Cloud infrastructures: AWS, Microsoft Azure, Google Cloud, Heroku, etc.

Mobile tech stack

Some programming languages and frameworks are only suitable for building applications for either an Android or iOS platform. Let’s take a deeper look at the technologies used to create mobile apps.

For Android:

1. Most common programming languages — Java, Kotlin
2. Development Tools — Android Studio, Android SDK
3. Most common UI frameworks — Jetpack Compose, Android UI, UI kit

For iOS:

1. Most common programming languages — Objective-C, Swift
2. Development Tools — Xcode, AppCode
3. Most common UI frameworks — UIKit, SwiftUI

Hybrid mobile app development entails producing mobile apps for several platforms at the same time. Such apps are created using a mix of web and native technologies. Developing an app for several platforms may provide your business with certain advantages, such as helping you reach a larger target audience, saving time and money on the development, increasing time-to-market, and others. The most common frameworks for hybrid mobile apps — Unity, React Native, Xamarin, and Flutter.

Having explored the basics of the technology stack, let’s examine the most popular tech stack on the market and their main characteristics.

Top 7 popular tech stacks for businesses that aim to succeed

It is no secret that choosing between an innovative technology stack and a time-tested one is recommended to choose the latter. Let’s look at a few popular, well-established tech stacks that are quite popular and credible among the IT community and business owners.

LAMP	Linux, Apache HTTP Server, MySQL, and PHP/Perl/Python	LAMP was one of the first open-source software-based stacks to gain widespread popularity. It is widely used among developers since it is adaptable to various web pages and apps. This stack is frequently used to save money on web development.
.NET	60+ frameworks and platforms are in the .NET stack, including CLI languages spread across 13 layers	Developers use .Net technology to construct a functionality and bug-free framework for dynamic and interactive online programs and apps.
MERN	MongoDB, React.js, Node.js, Express.js	The stack is designed to make the web app development process more fast and smooth. The stack is also quite efficient for startups since it's cost-effective, open-source, and very

		popular among top brands (e.g., UberEats, Walmart app).
MEAN	MongoDB, Express.js, AngularJS, and Node.js.	The MEAN stack's technologies are also free and open-source, with a friendly community where developers may seek assistance. The tech stack works great for startups and small businesses that aim to build dynamic web apps and websites.
Ruby on Rails	Works in tandem with JavaScript, HTML, and CSS	Great for startups that plan to develop database-backed web applications. It helps build easy-to-maintain, scalable applications. RoR development framework enables the creation of lightweight applications RoR comes with lots of libraries that make development rather time-efficient.
Java*	Spring, Spring Boot, Hibernate, Wildfly, Linux NGINX.	Java stack has huge community support and is considered one of the most reliable technologies. Enterprise solutions, banking applications, mobile apps, and much more can be developed with this stack.

React Native	Redux, Redux-Saga, Storybook, Jes, Redux Persist, and other	React Native is a mature and well-established technology for building cross-platform mobile apps. Developers prefer React Native because of its code clarity and simplicity. With the help of this stack, businesses can create mobile apps that provide a great native-like experience, increase time-to-market, and decrease development costs.

*Java is a rather comprehensive term. These are standard Java technologies that are usually meant by “Java tech stack.”

To summarize, we can say that the examined well-established tech stacks have more advantages over emerging ones since numerous top companies already prove their reliability. On top of that, they are supported and constantly updated by a giant community of tech professionals and enthusiasts.

A few words about the product. Since 2013, Slack has become a corporate messenger that has helped millions of people streamline communication and collaboration within companies. At the end of March 2020, Slack reported that the system was being used by a record 12 million people at the same time. Now let us take a deeper look at the technologies that aided in the app’s success.

1. **Programming Languages** — PHP, Java, JavaScript, Kotlin
2. **Libraries & Frameworks** — Spark, Electron, NodeJS
3. **Databases & Cloud Services** — MySQL, Redis, Amazon EC2, Fastly, Amazon CloudFront, Amazon Route 53

How has the tech stack contributed to the product's success? Thanks to Electron's cross-platform nature, this desktop program is available for Windows, macOS, and Linux. Slack relies on MySQL technology for high availability, scalability, operability, extensibility, and performance. Since security is one of the top priorities for Slack, constant monitoring of the infrastructure was possible through a centralized logging system enabled by the usage of StreamStash, Elasticsearch, and ElastAlert. In turn, scalability is achieved with the help of Kafka.

A few words about the product. Snapchat is a picture and video-based mobile messaging app. There are over 265 million active Snapchat users throughout the world. In the United States, 75% of Millennials and Generation Z are Snapchat users. This achievement was aided by using the correct technological stack; let's look deeper at these technologies.

1. **Programming Languages** — Java, Swift, Kotlin, Objective-C, JavaScript
2. **Libraries & Frameworks** — Cocoa Touch, JQuery, Angular JS, Bootstrap, React

3. **Databases & Cloud Services** — NoSQL, MySQL, DocumentDB, MongoDB, Redis

How has the tech stack contributed to the product's success? Snapchat uses Amazon Web Services (Route 53, CloudFront) for hosting. Google Compute Engine is used to improve automated scalability for a large number of users and to handle temporary data storage. NoSQL is the main database that can manage a high number of structured data while being flexible.

As you can see, the best tech stack can bring long-term success to your product. With the correct tech stack, you can provide your app with scalability, high-level security, and modern functionality and gain a competitive advantage.

Factors to consider when choosing the right technology stack: our tips

After reading the previous part of our article, you're undoubtedly wondering how to pick a tech stack that will help achieve the success of the popular products listed above? The answer is to know the list of factors that should be considered carefully in order not to choose your tech stack that doesn't fit the

requirements and specifics of your product. Thus, let's examine all of the important criteria for picking a tech stack.

Type of the product

When choosing the technology stack, it's important to take into account the type of product you aim to build, namely, its complexity, business objectives you aim to achieve with it, the load it's supposed to withstand, its size, and potential growth. The more complex the project is (e.g., if you want to make a social media app or an enterprise solution), the more sophisticated combination of tools it will require to implement its functionality.

Your financial capacity

Despite the fact that most of the tools are ready-made or open-source, product owners still need to consider several important aspects that can affect their budget: the developer's rate that varies depending on the specialization, licensing fees, and product's maintenance costs. Ready-made tools allow building products faster, however, may be limited in terms of flexibility when it comes to the implementation of custom features. Thus, affordable technologies may have their own risks.

In turn, when it comes to hiring developers, we recommend searching for those specializing in the most well-established and popular technologies since it will be easier to find a replacement for the developer in case of an emergency or necessity in expanding your existing team.

For instance, according to Stack Overflow, JavaScript is the most popular programming language, whereas WebAssembly is the least popular. JavaScript experts will be easier to come by, and you'll be able to acquire assistance for your application quickly and affordably.

Scalability

It is necessary to consider the growth possibilities and requirements before approving the technology stack. The function of a tech stack in an application's scalability is to handle the growing volume of users and load and enable the expansion of the functionality. For instance, if you want to create an IoT app, you should rather carefully choose the tech stack since scalability is one of the most important aspects to consider when developing such products.

You should select a technology stack that allows the addition of new compelling features and withstands unpredictable increases in user numbers. Here are some examples of popular scalable tech stacks and separate technologies: MEAN, MERN, RAD, Node.js, React, etc.

Strong security

It's no secret that security is critical at all times, and it's also essential to recognize that the risk of cyber attacks grows every year as technological progress grows. For example, if you are planning to build a video chat app, messenger, medical software, or any other solution that contains rather

sensitive user information, it is worth protecting yourself with a reliable text stack, excluding all the risks. For example, most popular frameworks like Node.js contain security guidelines.

So, keep in mind that it's highly crucial to adopt only secure and reliable tools that will help you enable the highest level of security to your solution and constantly update them to avoid open vulnerabilities, like the 0-day vulnerability that was discovered in the popular Java logging library log4j.

Maintenance and team expertise

It's important to determine whether or not your team will be able to maintain the application once it is deployed. This is the following step after you finish developing your app, and it's also linked to your tech stack selection. Unless you intend to cooperate with an outsourcing software development company that provides product maintenance services after its implementation, you should take into account your team's experience when choosing a tech stack. It will be better if you know for sure that your team will be able to use a tech stack successfully and doesn't require additional training; otherwise, there is no need to consider one.

What determines the cost of the tech stack?

So, now that we've covered the fundamentals of choosing a technology stack, it's time to turn our attention to the topic of money. We understand that startups are frequently on a limited budget. That's why, in this part, we'll show you what factors influence the price of a tech stack and provide recommendations on how to save money.

So what are the factors that affect the cost of a tech stack?

Popularity and maturity

The price of the technology stack is greatly influenced by it's maturity and popularity of specific technologies. The old and popular technologies are proven, reliable, and have a large pool of specialists to choose from. You can always find developers specializing in Java but finding Clojure developers will be incredibly difficult. Thus, the fewer specialists who know the new or "unpopular" technology, the more expensive their price tag is.

Therefore, it is always better to turn to the old school. However, we stress that technologies should be both popular and mature since solely the popularity of the project isn't the only aspect to rely on, while the maturity of certain technologies may sometimes go together with an insufficient pool of specialists to choose from.

Licensing

Certain technologies can be very expensive because licensed solutions often have subscriptions, which can include fees for traffic, CPU, and usage in general.

Cost of maintenance

Most companies that employ software systems spend a considerable part of their money on the system's maintenance since the work with the product doesn't end when it's released. When choosing a tech stack and hiring developers with the corresponding expertise, mind such expenditures. Product maintenance often includes spending on the solution's optimization, functionality expansion, bugs identification, fixing, etc. Hence, consider opting for cheaper open-source technologies that can be updated and changed without limitations.

How to save and gain?

We've explored the factors that affect the price of a teck stack. Now let's find out how you can save money.

1. **Find an experienced and proven software vendor**

   Choose a software vendor that will understand your needs, provide proactive solutions to support your vision, and be a trustworthy partner. The professional software company will also assist you in lowering the charges of your tech stack. They are experts in their field and will undoubtedly advise you on cost-cutting measures.

   For instance, when our customers opt for our services with the request to develop document management systems, we always recommend opting for the Alfresco Community Edition solution that has robust OOTB functionality sufficient to cover the essential document management needs and costs approximately $1,400 and $2,800

depending on the type of installation. Such an option is significantly cheaper than the custom implementation of a DMS. And this is just one of the possible scenarios in which an experienced software vendor can help you lower expenditures.

2. **Use ready-made solutions**

   Such a recommendation is more accurate for startups with limitations in terms of budget and for businesses who want to check their idea with MVP. When it comes to cost, ready-made solutions (e.g., third-party extensions, community-built tools, etc.) can substantially influence cost reduction since they help not to develop a significant part of functionality from scratch, which usually takes lots of time, and as a result money.

3. **Cross-platform tools for mobile development**

   If the goal of your startup is to create a competitive mobile app, but budget restrictions put lots of pressure, you can always save money by employing cross-platform tools instead of native technologies. You may choose the right technology for your application, which will help launch an app across several platforms to save resources in this way. One of the best cross-platform tools on the market in 2022 are React Native, Xamarin, Adobe PhoneGap, Ionic, Flutter, and Sencha.

Feedback vs Discovery phase

Feedback. When we say feedback about choosing a tech stack, we mean doing your own research on the market and basing your choice on users' feedback. Unfortunately, startups can often choose a technology stack based on users' feedback or the opinion of other business owners who have launched their products. But it's easy to forget that there's no one-size-fits-all solution, and just because one tech stack worked for another project doesn't mean it'll work for yours. Without a proper strategy, professional research, tech background, and many other nuances, it's simply impossible to determine the right stack for your project.

Discovery phase. As we all know, the stack isn't something that can be repaired with a wave of a magic wand after the product is released. So, when it comes to choosing a tech stack, we highly recommend not skipping one of the most important phases in software development called the discovery stage. This stage aims to avoid fatal errors, reduce possible risks, obtain clear technical descriptions, choose the most appropriate tech stack, and much more. Let's look closer at those deliverables of this phase that can help when choosing a technology stack.

1. **Requirements specification** — with an in-depth analysis of your requirements and their corresponding documentation, it's easier for a software architect to choose the right tech stack that will help build the project that meets your vision.
2. **Competitive analysis** — a profound examination of the direct competitors' products and the technology stack behind them can help evaluate tools that should be a part of your solution's stack.

3. **Feature list** — the document containing a list of the project's features. With this document, the team will be able to understand what kind of technology your project needs.

Now we can understand that shallow research and users' feedback who don't know all the subtleties of your project can only damage your project. But a team of professionals will gather all the essential information about your future product and, based on this information, will be able to choose the most acceptable text stack.

Our experience how to choose a tech stack for your project

We suggest taking a closer look at one successful project implemented by the Aimprosoft team.Examine how our team overcame challenges with the outdated tech stack and fulfilled the client's demands.

Backstory. Since 2013, Aimprosoft has been working on

Genesys PGR, an online platform for the Global Crop Diversity Trust. Genesys is an online database of Plant Genetic Resources for Food and

Agriculture (PGRFA) conserved in genebanks throughout the world. It's a smart and centralized single entry-point via which users can seamlessly access data from genebanks for various purposes.

Issues that needed to be resolved. The client approached us with a request to achieve the following objectives:

allow genebanks to exchange data about their crop collections;
enable breeders to generate new varieties that are resistant to pests, illnesses, and changing climatic factors using genebank material;
ensure that the platform's core is easy to access to genebank data;
the ecosystem of libraries and frameworks is constantly evolving. To stay relevant, Genesys must be kept up to date.

The selected tech stack.

Backend technologies — Eclipse Jetty, MariaDB, MySQL, Hibernate, Elascticsearch, Hazelcast, Asciidoctor, Amazon S3, Liquibase, QueryDSL, Spring, OAuth 2
Frontend technologies — React, Webpack 4, i18n, Redux, SCSS, d3.js, Material UI

Solution. Our development team used new technologies in the Genesys Catalog project, including React on the frontend and REST APIs on the backend. The objective was to transition to React because the frontend and backend were tightly connected in the old project. React and Angular was

considered for the implementation of sever-side rendering. Our team decided to continue further using React after the success of the Genesys catalog project.

In the new version, there is such a feature as visualization with interactive maps. Our developers used Leaflet, a JavaScript library for interactive maps, to create a map-based search for anyone looking to learn more about crops' provenance, which made the user experience more brilliant. For rapid data contribution, the Genesys platform provides APIs and permits the use of a Java-based Anno upload tool.

Outcome. The choice of technology was made on emerging and new technologies at the time, which today are among the most popular. Along with the development of technology, our team has improved the project. The project is constantly updated and adjusted to the requirements of the selected technology. The technology stack of this online platform has been working well for years, ensuring the project's success. Overall, our fruitful cooperation with the client and correctly chosen tech stack resulted in the creation of a top-notch solution of global importance that allows users to explore more than 4 million plant samples.

t is always the best idea to explore the available web development technologies before finalizing the tech stack to go with. From artificial intelligence programming to blockchain, many new innovations and

advancements are coming into full stack web app development field. Here are some of the latest full-stack web app development technologies that you must consider before starting your web development project. To give you the best idea, we will also mention the pros and cons of every tech stack.

MERN Stack

The MERN stack combines MongoDB, Express.js, React, and Node.js to create dynamic web apps.

Frontend:

- React: Powerful library for building user interfaces, enabling component-based development and efficient state management.

Backend:

- Node.js: Server-side runtime environment for executing JavaScript code, facilitating fast and scalable server-side development.
- Express.js: Lightweight web application framework for Node.js, providing robust routing and middleware capabilities.

Pros:

- Consistent language (JavaScript) throughout the stack, enhancing developer productivity and code maintainability resulting in quicker app deployment for your business.
- React's virtual DOM technology improves the performance of your application by minimizing reloads on the web app pages.
- Node.js enables asynchronous, event-driven I/O operations, enhancing scalability and performance of server-side code.
- Express.js simplifies server-side development with minimalistic and flexible architecture.

Cons:

- MERN has a Complex setup compared to traditional LAMP stack.
- If you do not hire experienced MERN stack developers, then handling complexities can be tough with this full stack development technology.

MEAN Stack

The MEAN stack comprises MongoDB, Express.js, Angular, and Node.js, offering a full JavaScript-based solution for web apps.

Frontend:

- Angular: Excellent front-end framework maintained by Google, offering powerful tools for building appealing web apps.

Backend:

- Node.js: An open source Java Script runtime environment for building robust web apps.
- Express.js: A lightweight and flexible framework great for startups.

Pros:

- Full JavaScript stack enables code reuse and enhances developer productivity.
- Angular’s two-way data binding simplifies UI development and synchronization with back-end data delivering great efficiency to your web app.
- Strong community support and extensive documentation for all components helping your MEAN app development team to effectively maintain the app.

Cons:

- Steeper learning and complexity curve, especially for Angular.
- Angular’s frequent updates may require frequent code changes.

LAMP Stack

The LAMP stack, consisting of Linux, Apache, MySQL, and PHP, is a traditional and widely used solution for web app development.

Frontend:

No specific frontend framework is there in the LAMP stack. Mostly LAMP Stack developers use HTML, CSS, and JavaScript.

Backend:

- Linux: Operating system.
- Apache: Web server software.
- MySQL: Relational database management system.
- PHP: Server-side scripting language.

Pros:

- Mature and stable technologies with a long history of usage in web development.
- Wide availability of hosting options and compatibility with various server environments.
- MySQL's relational database model is well-suited for structured data storage.

Cons:

- Not as scalable as NoSQL databases for handling large datasets or high traffic.

- PHP’s weak typing and inconsistent function naming can lead to code maintenance challenges in large projects.

Ruby on Rails (RoR)

Ruby on Rails, built on top of the Ruby programming language, emphasizes convention over configuration and promotes rapid development.

Frontend:

Ruby on Rails typically integrates with front-end frameworks like React or Angular, but can also use ERB (Embedded Ruby) templates for server-side rendering.

Backend:

- Ruby: Programming language emphasizing simplicity and productivity.
- Rails: Full-stack web application framework written in Ruby, emphasizing convention over configuration.

Pros:

- Rapid development with built-in conventions and generators.
- Active community and extensive gem library for added functionality.
- Ruby’s elegant syntax promotes clean and maintainable code.

Cons:

- Learning curve for developers unfamiliar with Ruby's syntax and Rails' conventions.
- Performance overhead compared to more lightweight frameworks.

Django

Django, a Python-based framework, offers a high-level approach to web development, emphasizing rapid development.

Frontend:

Django typically integrates with front-end frameworks like React or Angular, but can also use Django Templates for server-side rendering.

Backend:

- Python: Programming language known for its readability and simplicity.
- Django: High-level Python web framework that encourages rapid development and clean, pragmatic design.

Pros:

- Highly suggested for rapid development.
- Django ORM simplifies database interactions and protects against SQL injection.
- Python’s readability and extensive standard library enhance developer productivity.

Cons:

- This is an Opinionated framework which may limit flexibility in certain cases.
- Django’s synchronous nature may not be suitable for high-concurrency apps without additional optimization.

Vue.js

Vue.js is a progressive JavaScript framework for building user interfaces, offering simplicity, flexibility, and excellent performance.

Frontend

Vue.js can be used as a standalone frontend framework or integrated with backend technologies like Django and Node.js app development.

Backend

No specific backend technology hence it can be paired with various server-side frameworks or platforms.

Pros:

- Lightweight and easy to learn, suitable for both small and large-scale projects.
- Component-based architecture promotes reusability and maintainability.
- Excellent documentation and a growing community ecosystem.

Cons:

- Smaller ecosystem compared to React and Angular, with fewer third-party libraries and resources.
- Limited corporate backing compared to Angular (Google) and React (Facebook).

Latest Trends in Full-Stack Web Development

In the above section, we have seen the latest and greatest Full stack web development technologies. However, it is also crucial to keep up with the latest trends taking place in the full-stack web development domain. Here are

some of the latest trends and technologies that you might consider for your web project giving an idea about what is going on in the dynamic market.

Artificial Intelligence (AI)

AI development services are in high demand in the market as more businesses are looking forward to automating business operations. Integrating AI into your web project can enhance:

- User experiences
- Automate processes
- Improve decision-making

Machine Learning (ML)

ML algorithms enable systems to learn from data and make predictions. ML can be applied to achieve:

- Personalized recommendations
- Efficient data analysis
- Natural language processing
- Enhancing the features and functionality of app

Read More: How To Apply Machine Learning In Android App Development?

Blockchain

Blockchain technology ensures secure and transparent transactions, commonly associated with cryptocurrencies but with broader apps in supply chain management, identity verification, and decentralized applications.

Cybersecurity

With evolving cyber threats, cybersecurity solutions are crucial for web applications. Implementing encryption, authentication protocols, and regular security audits can safeguard sensitive data and protect against cyber attacks.

Cloud Computing

Cloud services offer scalability, flexibility, and cost-effectiveness for web development. Platforms like AWS, Azure, and Google Cloud provide infrastructure-as-a-service (IaaS) and platform-as-a-service (PaaS) solutions, enabling your business to focus on building and deploying applications without worrying about underlying infrastructure.

Augmented Reality (AR) and Virtual Reality (VR)

AR and VR technologies create immersive experiences for users, offering new opportunities for interactive content and engagement. Integrating AR and VR

into web projects can enhance user experiences and differentiate your web app in the market.

Read More: Building Augmented Reality (AR) Apps for Android: Tools and Frameworks

Progressive Web Apps (PWAs)

PWAs combine the best of web and mobile app experiences, offering offline functionality, fast loading times, and responsive design. PWAs are becoming popular for cross-platform development, providing a seamless user experience across devices.