

React: A detailed survey

Varun Komperla, Pratiba Deenadhayalan, Poonam Ghuli, Ramakanthkumar Pattar

Department of Computer Science and Engineering, R V College of Engineering, Mysore Rd, RV Vidyaniketan Post, Bengaluru, India

Article Info

Article history:

Received Jun 18, 2021

Revised Feb 4, 2022

Accepted Apr 13, 2022

Keywords:

JavaScript

JSX

React

React native

Single-page applications

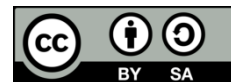
Virtual document object model

Web frameworks

ABSTRACT

With rapid developments in technology and the ever-growing number of web applications, it has become a necessity to create fast and scalable applications to cater to the current market. There are a plethora of frameworks available for web development, and a developer must choose the most appropriate framework for their use. In this paper, a detailed analysis of the history, prominent features and advantages of React, an open-source JavaScript library is presented. A discussion on React Native, a framework for building native applications is also given. This paper has provided an insight into the reason React is the leading web development framework in the world

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Varun Komperla

Department of Computer Science and Engineering, R V College of Engineering

Mysore Rd, RV Vidyaniketan Post

Bengaluru, India

Email: varunkomperla.cs17@rvce.edu.in

1. INTRODUCTION

In 2011, as Facebook was growing in size and popularity, its application was becoming more difficult to manage due to a large number of updates. Jordan Walke, a software engineer at Facebook, created a prototype of React called “FaxJS” [1] to make the process more efficient. He had taken inspiration from XHP, an HTML component library for PHP. React was first used in Facebook’s news feed in 2011, and later on Instagram in 2012. At JSConf US in 2013 [2] it was announced the React would be open-sourced, thus enabling everyone to develop smooth, interactive user interfaces for their applications. React Native, a framework that enables the development of native mobile applications for iOS and Android, was also created by Facebook and open-sourced in 2015. The most notable and differentiating feature of React was that of the virtual document object model (DOM) [3], which made it suitable for high performance applications. React also has a significant number of other advantages over other frameworks. It speeds up the development process due to its modular structure allowing developers to work in parallel on individual components, it’s flexible and easy to maintain thanks to self-contained components, it has been designed for high performance, and it allows for mobile development with React Native. Currently, React is one of the most popular JavaScript frontend frameworks with many popular organizations like Facebook, Instagram, Airbnb, Twitter, and Netflix using it extensively. In this paper, an in-depth analysis of React is presented.

2. RESEARCH

In this section, some related research and background has been presented. This will help to better understand the impact of the features that the React framework possesses. Some fundamental concepts of

React which make it so popular have also been described. An overview of React is presented, followed by the most popular features of React and finally, a short insight into React Native is given.

2.1. Related research

2.1.1. Single page applications

One needs to understand what a single-page application (SPA) is before one can truly appreciate the advantages of React. In the paper, Jhadav [4] an SPA is defined as a web application that is comprised of many individual components which can be replaced independently of each other without refreshing the entire page [5] thus eliminating the need to reload the page upon every user action. In the early days of the internet, the main purpose of traditional websites was to serve static pages to their clients [6]. For each user interaction, the entire webpage had to reload to reflect the result of the interaction. This resulted in slower response times, higher bandwidth usage and dissatisfaction amongst users. The solution to this is an SPA, where the initial page is loaded only once along with all of its resources like CSS, images, scripts, etc. and then only the appropriate components are updated dynamically based on user interactions. This causes every subsequent request, after the initial page is loaded, to take a significantly shorter amount of time since only a portion of the page is being refreshed rather than the entire page [7]. The generic Client-Server Request and Response interaction is shown in Figure 1.

2.1.2. Client-side routing

This raises the question, “How does one implement a single-page application?” client-side routing is the answer to this question. With client-side routing, routes are internally handled by a JavaScript file at the client-side and rendered to the front end. When a client clicks on an internal link, the URL is updated to show that there is an update happening to the page without reloading it. This provides users with a more intuitive sense of where they are within the application and also provides the ability to use deep links to load a specific view in the application. With client-side routing, only the newly required data is retrieved from the server and the appropriate portion of the page is dynamically updated with this data. There is no need to reload the page for every route in the application. It is important to understand that React can be used to build a variety of applications, with single-page applications being one of the more famous types of applications in the world today.

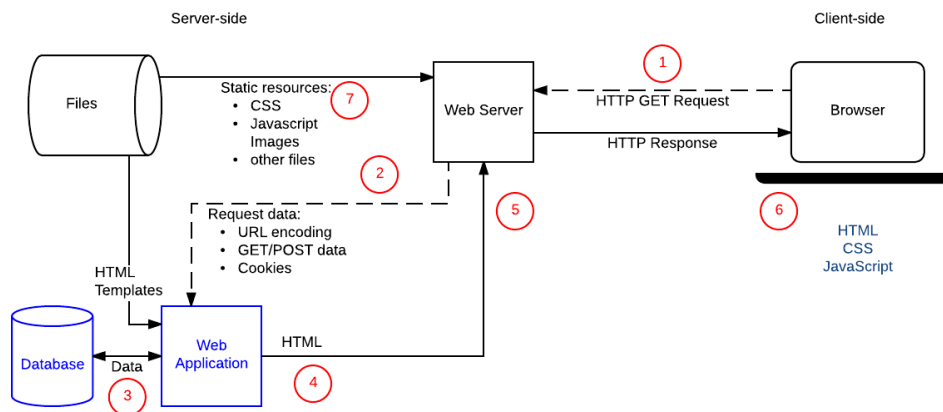


Figure 1. Client-server request and response interaction [8]

2.2. Overview of React

React is an open-sourced JavaScript library that is used to build interactive user interfaces (UIs), especially for single-page applications. The purpose of React is to build simple, fast, and scalable applications. React implements the view (V) part of the model-view-controller (MVC) architecture [9]–[13]. React gained popularity mainly due to its declarative and component-based nature [14]. One can create components that manage their own state and build more complex UIs by structuring these components together. React components are independent bits of code that enable re-usability. A component is conceptually similar to a JavaScript function. It accepts arbitrary inputs called “props” and returns some UI elements that will be rendered on the screen. With this design, a single component can be reused multiple times across various views within the application. React is a JavaScript library and thus supports

ECMAScript6 (ES6). React can also be rendered on the server-side using NodeJS. Furthermore, React can be used to power native mobile applications using React Native.

2.3. React features

React has many salient features that are the reason for its popularity. Some of these features are its simplicity, its component based nature and its support if JSX. There is also the virtual DOM which is the reason for improved performance of the user interface when using React, its capability of routing within an application and its nature of unidirectional data flow. All these features of React are described in the following sub-sections in more detail.

2.3.1. Simplicity

A very important reason why React is so popular among front end developers is due to its simplicity. One just needs to have an understanding of HTML and JavaScript. Developers migrating from another framework can easily adapt to React since it focuses only on building user interfaces.

2.3.2. Components

This is a key feature of React. Components are independent, reusable bits of code and are the building blocks of React code. They make it possible to split the UI into more manageable pieces and let one think about each piece in isolation. Data can be passed to components in the form of a “props” object which can be used to dynamically render elements in the DOM. Unlike the “props” object, which is passed from outside the component, each component has a “state” object that is managed within the component.

There are two types of components in React:

- Class component: a JavaScript class extending which has a `render()` method and extends the *React.Component* class.
- Functional component: a JavaScript function that returns JSX. They are also called stateless components since they do not possess their own state.

Each component also has its own lifecycle, consisting of three main stages—mount, update, and unmount. The component can be monitored and manipulated during various stages of its lifecycle using lifecycle methods. Some common React lifecycle methods are listed in Table 1. There are other lifecycle methods like *shouldComponentUpdate()*, *getDerivedStateFromProps()* and *getSnapshotBeforeUpdate()* but these are rarely used. The vast number of lifecycle methods provide great flexibility in manipulating components at various points in time.

Table 1. List of some React lifecycle methods [15]

Lifecycle Method	Description
<code>render()</code>	It is called during the mount and update phases of the component. For class components, it is a mandatory method. The state cannot be modified within <i>render()</i> .
<code>componentDidMount()</code>	It is called after the component has mounted and is ready. The state can be changed within this function. Any API calls are usually initiated within this function.
<code>componentDidUpdate()</code>	It is called after the component has updated, usually in response to changes in the “props” or “state” objects. The state can be changed within this function.
<code>componentWillUnmount()</code>	It is invoked before the component is unmounted. Clean-up operations like aborting network calls, destroying objects that were created, etc. are usually carried out in this function. Since this is called just before a component is unmounted, the state cannot be modified.

2.3.3. JSX

JSX is what makes React a powerful library. JSX is an extension to JavaScript [16], analogous to how XHP is an extension of PHP. It allows one to write HTML within JavaScript. It enables the structuring of components in HTML, which most web developers are familiar with. JSX is used to create React elements. React components return elements to be rendered on the UI using JSX. For example:

```
const Hello = <h1> Hello World! </h1>
```

as can be seen in the example, the HTML `<h1 />` element is stored in a constant string which is using JSX syntax. JSX also prevents cross-site-scripting (XSS) attacks.

2.3.4. Unidirectional data flow

React uses one-way data bindings. This means that in React, data can only flow in a single direction. So data has only one way in which it can be passed to various parts of the application. As shown in Figure 2,

the state is passed to the view, that is, to a component and its children. When the state needs to be updated, an action is called by the view. The state is changed as a result of an action and is passed again to the view and its child components. As a result of one-way bindings, data cannot flow in the opposite direction (as opposed to two-way bindings in Angular). For example, the view cannot directly change its state. It must trigger an action to do so. A state is owned by a single component in React and any change in the state of a component can only affect its child components below it.

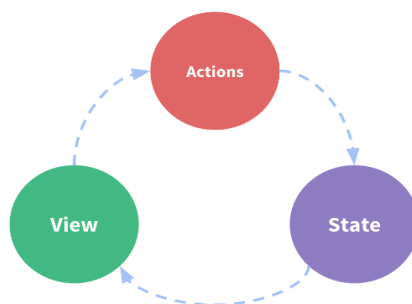


Figure 2. Data flow in React [17]

2.3.5. Virtual DOM

Traditionally, whenever an update needs to be made to an element in the DOM, the following steps occur:

- The browser parses the HTML to get the DOM.
- The required element is extracted from the DOM tree.
- The DOM is updated with the new value.
- The CSS is applied again to the parent and child, which involves calculations.
- The coordinates of each element in the layout are updated.
- Finally, traverse and render the DOM tree on the browser.

As seen in the steps above, updating the DOM has a lot of other operations attached to it. This is why it is a slow process. React maintains an in-memory data structure called a virtual DOM [3], [18] which is a lightweight representation of the actual DOM. Whenever a change is made to an element, the Virtual DOM gets created again, which is a fast process since nothing gets rendered on the browser. React maintains two Virtual DOMs at a time, one is the updated Virtual DOM and the other is the version of the Virtual DOM before the update. It uses a “diffing” algorithm [19] to compare the two Virtual DOMs and see what has changed after the update. React then applies only the required changes to update the real DOM in the minimum number of steps. This improves the performance [20], [21] of the user interface significantly and is the reason why React is desirable by developers around the world.

2.3.6. Routing

React router is the standard routing library used in React [22]. The standard way of introducing routing into a React application is to have the App component return a router component [23], which will wrap the Switch component containing all of the route components. An example of basic routing syntax is shown in Figure 3. The Switch wraps all the route components and renders the first route whose path matches the current URL. Each route component configures the route and wraps the components that will render for the configured route.

2.4. React Native

React Native is an open-source framework that is used to build native mobile applications. It can be used to develop applications for Android and iOS. React Native is popular because it is easy for developers to adapt to it, since it uses JavaScript—one of the most famous programming languages out there along with the regular web technologies (HTML). The difference is that applications developed in React Native compile to native code. Any developer with an understanding of React can pick up React Native as well, since they are very similar. The major appeal is that developers would be using familiar technologies to build mobile applications, which would otherwise require different domain knowledge. Another advantage of React Native is that most of the React Native APIs are cross-platform [24]–[27]. This means that an application will have consistent behaviour across Android and iOS platforms, and developers do not have to write code multiple

times for the same logic [28], [29]. If the need for platform-specific features arises, then React Native allows the use of platform-specific versions of each component as well. Thus, even with the use of platform-specific APIs, the portion of reusable code remains substantial [30]–[34].

```
function App() {
  return (
    <Router>
      <div>
        <nav>
          <ul>
            <li> <Link to="/"> Home </Link> </li>
            <li> <Link to="/about"> About </Link> </li>
            <li> <Link to="/docs"> Docs </Link> </li>
          </ul>
        </nav>

        <Switch>
          <Route path= "/about">
            <About/>
          </Route>
          <Route path= "/docs">
            <Docs/>
          </Route>
        </Switch>
      </div>
    </Router>
  );
}
```

Figure 3. Basic routing application in React

3. METHOD OF TESTING

Unit Testing is an important stage of any software development project [35]–[41]. This applies to building applications in React as well. Some advantages of unit testing are:

- Improvement in design of solutions.
- Improvement in code quality.
- Aids the debugging process.
- Reduction in the cost of development.

Given the benefits of unit testing as mentioned above, there are two tools recommended by the developers of React, for testing. These are jest and React testing library [42]. Jest is a test runner in JavaScript that allows access to the DOM through jsdom. Jest can be used to mock React components without their children, and provides more control over the execution of the code. React Testing Library comprises of a set of helpers that enables the testing of react components while abstracting their implementation details. It allows for easy refactoring of code and also encourages developers to adopt the best accessibility practices. Since React is based on JavaScript, possessing extensive knowledge of testing tools [43], [44] can help programmers design simple, efficient and reliable interfaces using React as there is an abundance of testing facilities available for developers.

4. RESULTS AND DISCUSSION

In this section, the results of the survey have been presented concisely, along with a discussion. The advantages of React, which are the reasons for its success as the leading framework for front end development, have been listed in brief. This is followed by a discussion summarizing the motivation behind React, its features and the present state of the framework as well as its future.

4.1. Advantages of react

After looking at the various features of React, it is clear that React provides many advantages. These advantages are listed:

- With JSX, the process of writing components becomes much easier. It allows for simpler and cleaner source code while building custom components.
- With the Virtual DOM, React allows for faster rendering [45], quick application load times and a better user experience.
- Due to the unidirectional flow of data, React also ensures stable code.
- With React Native, the option to build native applications is available, as well as the capability to build cross-platform applications (iOS and Android) .
- React also comes with a developer toolset that helps in the design and debugging processes.
- It boosts productivity. The ability to reuse system components is one of the best features of React, due to which updates for an application are relatively simple to develop.
- It is simple to set up server-side scripting in React, thus making it SEO [46]–[51] friendly.
- With React Routing, the code can be split up into logical segments which enables better management of the code.

4.2. Discussion

React was developed with the aim of building large scale applications whose data repeatedly changes with time, and it has addressed this well. React allows for easy creation of interactive UIs, supports JSX, has a component-based structure, and is much faster due to the virtual DOM. These features are attributed to the success of React as a front end framework. The latest version of React is React v17.0 and was released three years after the previous version. While it did not introduce any new developer features, the development team claimed that it is an incredibly important stepping stone for the future of React. It focuses on easing the process of upgrading the React version of an application. Different parts of the application can be upgraded to the latest React version gradually, instead of updating the whole application at once. It also introduces a new JSX Transform, which enables the use of JSX without React. The next version of React to be released is v18.

5. CONCLUSION

The React community has significantly grown over the last few years and has become the most widely used front end framework. React faces competition from other web development frameworks such as Angular and VueJS. With this in mind, and with the increasing popularity and user base of web applications, the React community needs to keep updating the React library according to the requirements of the modern web developer to remain the preferred front end development framework. In this paper, a thorough analysis of the history and overview of React is presented along with a detailed explanation of the salient features that make React unique and widely used in the industry. Finally, a brief discussion on testing and the tools available in React is presented and the paper is concluded with the advantages and a discussion of the present situation of React.

REFERENCES





- [1] J. Walke, "FaxJS." github.com . <https://github.com/jordwalke/FaxJs> (accessed May 22, 2021).
- [2] J. Occhino, T and Walke, "JS Apps at Faebook." <https://www.youtube.com/watch?v=Gw0rj4sNH2w> (accessed May 22, 2021).
- [3] "Virtual DOM and internals." reactjs.org. <https://reactjs.org/docs/faq-internals.html> (accessed Dec. 28, 2021).
- [4] M. A. J. #1, B. R. Sawant, and A. Deshmukh, "Single Page Application using AngularJS," Accessed: Apr. 14, 2022. [Online]. Available: <http://mydomain.com/myseo#key=value>.
- [5] H. Han, Y. Xue, K. Oyama, and Y. Liu, "Practice and evaluation of pagelet-based client-side rendering mechanism," *IEICE Transactions on Information and Systems*, vol. E97-D, no. 8, pp. 2067–2083, 2014, doi: 10.1587/transinf.E97.D.2067.
- [6] M. Han and C. Hofmeister, "Modeling request routing in web applications," in *Proceedings of the Eighth IEEE International Symposium on Web Site Evolution, WSE 2006*, Sep. 2006, pp. 103–110, doi: 10.1109/WSE.2006.14.
- [7] W. Stepniak and Z. Nowak, "Performance analysis of SPA web systems," in *Advances in Intelligent Systems and Computing*, vol. 521, Springer International Publishing, 2017, pp. 235–247.
- [8] "Mozilla Developer Network (MDN) Docs," Accessed: May 22, 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Client-Server_overview.
- [9] A. Leff and J. T. Rayfield, "Web-application development using the Model/View/Controller design pattern," in *Proceedings - 5th IEEE International Enterprise Distributed Object Computing Conference*, 2001, vol. 2001-Janua, no. January, pp. 118–127, doi: 10.1109/EDOC.2001.950428.
- [10] M. Abrams, C. Phanouriou, A. L. Batongbacal, S. M. Williams, and J. E. Shuster, "UIML: An appliance-independent XML user interface language," *Computer Networks*, vol. 31, no. 11, pp. 1695–1708, May 1999, doi: 10.1016/S1389-1286(99)00044-4.
- [11] B. Bennett *et al.*, "A distributed object oriented framework to offer transactional support for long running business processes," in *IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing*, 2000, pp. 331–348.
- [12] K. Betz, A. Leff, and J. T. Rayfield, "Developing highly-responsive user interfaces with DHTML and servlets," in *IEEE International Performance, Computing and Communications Conference, Proceedings*, 2000, pp. 437–443, doi: 10.1109/pccc.2000.830348.

- [13] G. E. Krasner and S. T. Pope, "A Cookbook for Using the Model-View-Controller User-Interface Paradigm in Smalltalk-80," pp. 26-49, 1988.
- [14] "React," [Online]. Available: <https://reactjs.org/> (accessed 22 May 2021).
- [15] "React.Component." <https://reactjs.org/docs/react-component.html> (accessed May 22, 2021).
- [16] "Introducing JSX." <https://reactjs.org/docs/introducing-jsx.html> (accessed Dec. 28, 2021).
- [17] "Redux Fundamentals, Part 2: Concepts and Data Flow." <https://redux.js.org/tutorials/fundamentals/part-2-concepts-data-flow> (accessed May 22, 2021).
- [18] G. Psaila, "Virtual DOM: An efficient virtual memory representation for large XML Documents," in *Proceedings - International Workshop on Database and Expert Systems Applications, DEXA*, Sep. 2008, pp. 233–237, doi: 10.1109/DEXA.2008.117.
- [19] "Reconciliation." <https://reactjs.org/docs/reconciliation.html> (accessed Dec. 28, 2021).
- [20] D. Chęć and Z. Nowak, "The performance analysis of web applications based on virtual DOM and reactive user interfaces," in *Advances in Intelligent Systems and Computing*, vol. 830, Springer International Publishing, 2019, pp. 119–134.
- [21] E. Czaplicki and S. Chong, "Asynchronous functional reactive programming for GUIs," in *Proceedings of the 34th {ACM} {SIGPLAN} conference on Programming language design and implementation - {PLDI} {textquotesingle}13*, 2013, p. 411, doi: 10.1145/2491956.2462161.
- [22] "React Router: Main Concepts," [Online]. Available: <https://reactrouter.com/docs/en/v6/getting-started/concepts> (accessed 28 12 2021).
- [23] V. Subramanian, "Routing with react router," in *Pro MERN Stack*, Berkeley, CA: Apress, 2017, pp. 151–171.
- [24] M. Martinez and S. Lecomte, "Towards the quality improvement of cross-platform mobile applications," in *Proceedings - 2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems, MOBILESoft 2017*, May 2017, pp. 184–188, doi: 10.1109/MOBILESoft.2017.30.
- [25] A. Charland and B. LeRoux, "Mobile application development: Web vs. native," *Communications of the ACM*, vol. 54, no. 5, pp. 49–53, May 2011, doi: 10.1145/1941487.1941504.
- [26] R. Francese, M. Risi, G. Tortora, and G. Scanniello, "Supporting the development of multi-platform mobile applications," in *Proceedings of IEEE International Symposium on Web Systems Evolution, WSE*, Sep. 2013, pp. 87–90, doi: 10.1109/WSE.2013.6642422.
- [27] J. Perchat, M. Desertot, and S. Lecomte, "Component based framework to create mobile cross-platform applications," *Procedia Computer Science*, vol. 19, pp. 1004–1011, 2013, doi: 10.1016/j.procs.2013.06.140.
- [28] M. E. Joorabchi, A. Mesbah, and P. Kruchten, "Real challenges in mobile app development," *International Symposium on Empirical Software Engineering and Measurement*, Oct. 2013, pp. 15–24, doi: 10.1109/ESEM.2013.9.
- [29] E. Angulo and X. Ferre, "A case study on cross-platform development frameworks for mobile applications and UX," in *ACM International Conference Proceeding Series*, 2014, vol. 10-12-Sep, doi: 10.1145/2662253.2662280.
- [30] T. A. Majchrzak, A. Biørn-Hansen, and T. M. Grønli, "Comprehensive analysis of innovative cross-platform app development frameworks," in *Proceedings of the Annual Hawaii International Conference on System Sciences*, 2017, vol. 2017-Janua, pp. 6162–6171, doi: 10.24251/hicss.2017.745.
- [31] H. Heitkötter, H. Kuchen, and T. A. Majchrzak, "Extending a model-driven cross-platform development approach for business apps," *Science of Computer Programming*, vol. 97, no. P1, pp. 31–36, Jan. 2015, doi: 10.1016/j.scico.2013.11.013.
- [32] J. Ohrt and V. Turau, "Cross-platform development tools for smartphone applications," *Computer*, vol. 45, no. 9, pp. 72–79, Sep. 2012, doi: 10.1109/MC.2012.121.
- [33] I. Dalmasso, S. K. Datta, C. Bonnet, and N. Nikaein, "Survey, comparison and evaluation of cross platform mobile application development tools," in *2013 9th International Wireless Communications and Mobile Computing Conference, IWCMC 2013*, Jul. 2013, pp. 323–328, doi: 10.1109/IWCMC.2013.6583580.
- [34] M. Palmieri, I. Singh, and A. Cicchetti, "Comparison of cross-platform mobile development tools," in *2012 16th International Conference on Intelligence in Next Generation Networks, ICIN 2012*, Oct. 2012, pp. 179–186, doi: 10.1109/ICIN.2012.6376023.
- [35] E. Daka and G. Fraser, "A survey on unit testing practices and problems," in *Proceedings - International Symposium on Software Reliability Engineering, ISSRE*, Nov. 2014, pp. 201–211, doi: 10.1109/ISSRE.2014.11.
- [36] A. Arcuri, G. Fraser, and J. P. Galeotti, "Automated unit test generation for classes with environment dependencies," in *ASE 2014 - Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*, Sep. 2014, pp. 79–89, doi: 10.1145/2642937.2642986.
- [37] G. Fraser, M. Staats, P. McMinn, A. Arcuri, and F. Padberg, "Does automated white-box test generation really help software testers?," in *2013 International Symposium on Software Testing and Analysis, ISSTA 2013 - Proceedings*, Jul. 2013, pp. 291–301, doi: 10.1145/2483760.2483774.
- [38] G. Fraser and A. Arcuri, "Sound empirical evidence in software testing," in *Proceedings - International Conference on Software Engineering*, Jun. 2012, pp. 178–188, doi: 10.1109/ICSE.2012.6227195.
- [39] G. Fraser and A. Arcuri, "Whole test suite generation," *IEEE Transactions on Software Engineering*, vol. 39, no. 2, pp. 276–291, Feb. 2013, doi: 10.1109/TSE.2012.14.
- [40] V. Garousi and J. Zhi, "A survey of software testing practices in Canada," *Journal of Systems and Software*, vol. 86, no. 5, pp. 1354–1376, May 2013, doi: 10.1016/j.jss.2012.12.051.
- [41] B. Robinson, M. D. Ernst, J. H. Perkins, V. Augustine, and N. Li, "Scaling up automated test generation: Automatically generating maintainable regression unit tests for programs," in *2011 26th IEEE/ACM International Conference on Automated Software Engineering, ASE 2011, Proceedings*, Nov. 2011, pp. 23–32, doi: 10.1109/ASE.2011.6100059.
- [42] "Testing Overview." [reactjs.org. https://reactjs.org/docs/testing.html](https://reactjs.org/docs/testing.html) (accessed May 22, 2021).
- [43] S. Artzi, J. Dolby, S. H. Jensen, A. Møller, and F. Tip, "A framework for automated testing of JavaScript web applications," in *Proceedings - International Conference on Software Engineering*, May 2011, pp. 571–580, doi: 10.1145/1985793.1985871.
- [44] Y. Zou, Z. Chen, Y. Zheng, X. Zhang, and Z. Gao, "Virtual DOM coverage for effective testing of dynamic web applications," in *2014 International Symposium on Software Testing and Analysis, ISSTA 2014 - Proceedings*, 2014, pp. 60–70, doi: 10.1145/2610384.2610399.
- [45] Y. K. Xing, J. P. Huang, and Y. Y. Lai, "Research and analysis of the front-end frameworks and libraries in e-business development," in *ACM International Conference Proceeding Series*, 2019, pp. 68–72, doi: 10.1145/3313991.3314021.
- [46] V. N. Gudivada, D. Rao, and J. Paris, "Understanding Search-Engine Optimization," *Computer*, vol. 48, no. 10, pp. 43–52, Oct. 2015, doi: 10.1109/MC.2015.297.
- [47] B. Xing and Z. Lin, "The impact of search engine optimization on online advertising market," in *ACM International Conference Proceeding Series*, 2006, pp. 519–529, doi: 10.1145/1151454.1151531.





- [48] J. B. Killoran, "How to use search engine optimization techniques to increase website visibility," *IEEE Transactions on Professional Communication*, vol. 56, no. 1, pp. 50–66, Mar. 2013, doi: 10.1109/TPC.2012.2237255.
- [49] M. Cui and S. Hu, "Search engine optimization research for website promotion," in *Proceedings - 2011 International Conference of Information Technology, Computer Engineering and Management Sciences, ICM 2011*, Sep. 2011, vol. 4, pp. 100–103, doi: 10.1109/ICM.2011.308.
- [50] Y. Chen and C. He, "Paid placement: advertising and search on the internet," *Economic Journal*, vol. 121, no. 556, pp. F309–F328, Oct. 2011, doi: 10.1111/j.1468-0297.2011.02466.x.
- [51] Y. Liu *et al.*, "Identifying web spam with the wisdom of the crowds," *ACM Transactions on the Web*, vol. 6, no. 1, pp. 1–30, Mar. 2012, doi: 10.1145/2109205.2109207

BIOGRAPHIES OF AUTHORS







Varun Komperla     has completed his Bachelor of Engineering from the department of Computer Science and Engineering, RV College of Engineering. His research interests are in the area of front end and back end software development. He can be contacted at email: varunkomperla.cs17@rvce.edu.in.







Dr. Pratiba Deenadhayalan     is working as an Assistant professor in Computer Science and Engineering department at RVCE. She received her Ph.D degree from VTU. She has published over 40 research papers. She has worked on various research and consultancy projects sponsored by Cisco, Citrix and Samsung. She can be contacted at email: pratibad@rvce.edu.in.



Dr. Poonam Ghuli     is an Associate Professor at the Department of Computer Science and Engineering, RV College of Engineering, Bengaluru. She has completed her PhD in the area of Machine Learning from Visvesvaraya Technological University, Karnataka, India in the year 2017. Her research interests are Data Analytics, Machine Learning and Computer Vision. She is actively involved in research and consultancy work supported by many leading industries. She can be contacted at email: poonamGhuli@rvce.edu.in.



Dr. Ramakanthkumar Pattar     is a Professor and HOD in the Computer Science and Engineering department at RVCE. His research interests are Digital Image Processing, Pattern Recognition and Natural Language processing. He has published over 100 research papers. He has executed several funded research and consultancy projects sponsored by DRDO, ISRO, AICTE, GE India Pvt.Ltd, CABS, HP, Nihon Communication Solutions Pvt.Ltd etc. He can be contacted at email: ramakanthkp@rvce.edu.in