React Native for Android: Cross-Platform Mobile Application Development

Sreekanth Dekkati[1], Karu Lal[2*], Harshith Desamsetti[3]

[1]Assistant Vice President (System Administrator), MUFG Bank, Arizona, USA
[2]Integration Engineer, Ohio National Financial Services, USA
[3]Software Engineer, Tata Consultancy Services (TCS), USA

*Corresponding Contact:
Email: karu021984@gmail.com

Manuscript Received: 24 Sept 2019 - Revised: 15 Dec 2019 - Accepted: 29 Dec 2019

ABSTRACT

React Native has been embraced by many companies, and some of the most well-known applications in the world, including Facebook, Uber, and Pinterest, are among those that have done so. Why is it so widespread? What does it mean to react natively? It will likely attribute a significant portion of its success to the company's ability to utilize a single codebase compatible with several platforms. The use of React Native is continually gaining more widespread support. While developing mobile applications, using React Native can have positive and negative effects, and this article will help us grasp both aspects. This study will provide all the necessary information before we use React Native in one of our applications. With this knowledge, we can determine if react native is the appropriate choice for our next project.

Key Words: React Native, ReactJS, Android Apps, iOS, NativeDriver

INTRODUCTION

Responding to the question, what does react native? React Native, a popular JavaScript-based mobile app framework, lets us build native iOS and Android apps. The framework enables us to build cross-platform apps using the same codebase. Based on React, it brings its beauty to mobile app development. Facebook released React Native as open-source in 2015. Within a few years, it became the top mobile development tool. React Native powers Instagram, Facebook, and Skype, among other famous mobile apps (Krill, 2015). JavaScript provides the framework for React Native, which allows developers to write genuine mobile applications that run natively on iOS and Android devices (Bodepudi et al., 2019). It is built on React, a JavaScript package that Facebook uses to construct user interfaces; however, rather than targeting browsers, it targets mobile platforms. To put it another way, web developers can now create mobile applications with a look and feel that is genuinely "native," and they can do it from the convenience of a JavaScript library that we are already familiar with and adore (Thodupunori & Gutlapalli, 2018). In addition, the majority of the code that we create may be shared between platforms, which makes it simple to develop applications for both Android and iOS at the same time using React Native (Aggarwal, 2018).

React Native applications are developed using a combination of JavaScript and a markup language similar to XML, known as JSX. This is quite similar to how React is written for the web. The "bridge" component of React Native then calls the native rendering APIs written in Objective-C (for iOS) or Java (for Android). This takes place behind the scenes (Mandapuram et al., 2018). This application will render using actual mobile user interface components rather than reviews; as a result, it will have the same appearance and feel as any other mobile application. React Native apps will be able to access platform features such as the user's location or the camera on the user's phone since React Native makes JavaScript interfaces available for platform APIs (Fedosejev et al., 2016).

Currently, React Native is compatible with both iOS and Android, and it can extend its support to additional platforms in the future (Mandapuram & Hosen, 2018). Both the iOS and Android operating systems will be covered in this book. The great bulk of the code that we build will be compatible with several operating systems (Dekkati & Thaduri, 2017). And the answer is yes: we really can utilize React Native to build mobile applications that are ready for production! An interesting anecdote: It is already being used in production for user-facing applications by companies like Facebook, Palantir, and TaskRabbit.

Several variables explain React Native's popularity (Paul & Nalwaya, 2016). First, React Native lets businesses power their Android and iOS apps with one line of code. This saves time and resources significantly. Second, React, a popular JavaScript library, was used to create the React native framework. Thirdly, the framework lets frontend developers build complex, production-ready mobile apps instead of web-based ones.

REACT NATIVE ARCHITECTURE

React Native's architecture allows cross-platform functionality. JavaScript may execute arbitrary native code via the Native Module framework, which exposes native class instances as JavaScript objects. Native iOS modules are written in Objective C or Swift, while Android modules are in Java or Kotlin. React Native uses platform-specific APIs to render native components. React Native renders iOS UI components using Objective C or Swift APIs. Android apps will use Java or Kotlin. When creating React Native apps, developers rarely write native code. Developers do not need Objective C or Kotlin to build React Native apps.

The JavaScript Virtual Machine (JavaScript Bundle) runs all JavaScript code in React Native apps. In iOS and Android simulators and devices, React Native leverages JavaScriptCore. Mobile devices can run JavaScript code using JavaScriptCore. The OS provides this framework on iOS devices. Android does not have JavaScriptCore; thus, React Native bundles it with the app. This marginally increases the Android app size. The gadget runs JavaScript programming using JavaScriptCore (Pinto & Coutinho, 2018). Chrome will run JavaScript code for debugging. Chrome communicates with the source code using WebSockets and V8. Notably, the V8 engine and JavaScriptCore are different contexts, and mistakes can only occur when the debugger is attached to a mobile device (Deming et al., 2018). React Native architecture revolves around the bridge. React Native Bridge is C++/Java (Mandapuram, 2017b). The bridge converts JavaScript to native code and vice versa. It converts JavaScript to platform-specific parts. The bridge receives JavaScript calls and uses Objective C, Swift, Kotlin, or Java APIs to show the application as before. Be aware that React Native performs all layouts on separate threads. React Native has three lines: JavaScript, Shadow, and Main.

In the JavaScript thread, logic decides what to display. API calls and application JavaScript code are executed here. The Shadow thread runs JavaScript thread actions in the background

(Kowalczyk & Plechawska-Wójcik, 2016). Here, the app's layout is calculated and sent to the interface. Only the main thread can update the UI. Hence, it is called the UI thread. The Bridge system displays the app on the device using React. The process does not affect user experience because asynchronous calls occur separately from the Main thread. React Native applications employ the Main thread for native Android or iOS user interface rendering.

The Main thread and JavaScript thread transmit asynchronous JSON messages to communicate. Asynchronous JSON messaging is efficient. Both threads depend on the bridge (Mandapuram, 2017a). Bridge transfers data to the Shadow thread from the JavaScript thread. The JavaScript thread serializes data as JSON and sends it as a string. The same occurs while transferring data from the Shadow to the Main Thread.

FEATURES OF REACT NATIVE

After discussing the React Native-built products, let us move on to the advantages. What exactly is React Native, and what are its main benefits while developing a mobile application?

- Cross-Platform Development: The most significant advantage is the ability to reuse code. This demonstrates to product owners that their software can work effectively on multiple platforms, which they truly value. They can reuse the same code for both operating systems if they incorporate ninety percent of the React Native framework. It is also possible to use the source code of the online application to construct a mobile application if both projects use the React Native framework (Mandapuram, 2016). This is yet another outstanding feature. In addition, the development time is cut in half thanks to the fact that it makes use of pre-developed components, all of which are contained within the open-source library.
- A Large Community of Developers: React Native is a free and open-source JavaScript framework that allows programmers to contribute their knowledge to the development of the framework. If a developer is working on an app and runs into problems, they can ask for help from the community. There is always going to be someone available to help them find answers to the questions that they have about their concerns.
- Increased Cost-Effectiveness: One of the advantages of developing with React Native is an increase in cost-effectiveness. This is because, as was said before, programmers can design applications for both iOS and Android using the same code. It indicates that we will not need to hire different development teams to work for iOS and Android to finish our project. Instead, we will only need a small group to get it done. The production of mobile applications using the React Native programming language is substantially more cost-effective than alternative programming languages that do not offer cross-platform development.
- Fast Refresh: Thanks to quick refresh, developers can continue to use the application even when installing new updates and making UI modifications. Because the modifications are immediately implemented, the programmer does not need to redo all their work on the software. Programmers can increase their output and reduce the time spent on compilation because any modifications made to the software do not result in the loss of any current state (Gutlapalli, 2017a).
- A Straightforward User Interface (UI) React Native development creates the UI of an app by utilizing React JavaScript. This makes the program more responsive and faster, reducing the time it takes to load and improving the user experience. Because it adheres to a

component-based design philosophy and has a reactive user interface, the framework is ideally suited for developing applications accommodating simple and complex designs.

- Applications That Dash: Some people believe React Native code can harm an app's performance. The difference in speed that may be seen between native code and JavaScript is almost invisible to the human eye.
- Ready for the Future: Given how rapidly the framework dominates the market and how simply it tackles development challenges, the future of cross-platform apps built with React Native appears to hold great promise. Even while it has a few drawbacks, which we will go over in the following part, it makes up for them thanks to its rapid expansion and user-friendliness.

ADVANTAGES OF REACT NATIVE

React Native stands out from Cordova and Ionic because it uses its host platform's normal rendering APIs. Webviews are used to render JavaScript, HTML, and CSS mobile apps. Although effective, this strategy has limitations, particularly in performance. Additionally, they rarely access the host platform's native UI elements. These frameworks frequently "feel" odd when they try to emulate native UI elements because reverse-engineering animation details are time-consuming and quickly outdated (Gutlapalli, 2017b).

React Native uses the platform's view rendering methods to convert our HTML into native UI elements. React runs independently from the main UI thread, so our project can perform well without sacrificing capability (Hitz et al., 2017). React Native re-renders views when props or states change, like React. Since React Native uses its host platform's UI libraries instead of HTML and CSS markup, it differs from React in the browser.

Developers familiar with React can write mobile apps with the performance, look, and feel of a native app using everyday technologies. In developer experience and cross-platform development, React Native outperforms smartphone development.

- **Code Reusability:** React Native's main advantage is its ability to create for several platforms simultaneously. Use a single code base across platforms for faster app development and speed to market, simpler and cheaper maintenance, and easier engineer onboarding. If both use React Native, web and mobile apps may share code.
- **Hot Reloading:** Developers can inspect real-time code changes without refreshing. This seemingly slight improvement may improve development and productivity by providing immediate feedback on code changes. Just reloading code parts can speed up entire compilations.
- **Performance:** React Native's "bridge" concept represents a breakthrough in cross-platform programming. Natively built code makes React Native apps faster than web-based cross-platform solutions. React Native claims "native-like" performance. However, it is more like a "near-native" performance.
- **Cost Efficiency:** Cross-platform development is often done for cost-effectiveness. The project usually requires a more minor team because code can be shared across platforms. It takes two teams to give two identical operational solutions, unlike native development.
- **React Native Developer Community:** The open-source community is growing. Facebook is continually expanding and improving the framework; their cooperation is

crucial. Even if we stumble across a problem that React Native has not addressed, we can find a network of people willing to help since they want to make the framework more robust and reliable.

- **Constantly evolving:** React Native is a young framework gaining customer acceptance. This suggests it is continuously changing and improving with new additions and upgrades. We may see faster, more effective app development. It is ideal for fast-paced mobile app development due to its dynamic environment.

RISKS AND DRAWBACKS OF REACT NATIVE

Utilizing React Native does come with some drawbacks, just like using any other technology does; nevertheless, whether or not it is a suitable fit for our team depends primarily on the specifics of our business. Because the project is still in its infancy, the most significant danger is arguably React Native's lack of maturity. Support for iOS was made available in March 2015, and support for Android was made available in September 2015. There is unquestionably potential for advancement in the documentation, which is also undergoing ongoing development. Several iOS and Android features aren't supported, and the community is still working to identify the most effective procedures (Sengupta et al., 2016). The good news is that we can, in the overwhelming majority of instances, create support for missing APIs, which we will cover.

Because it adds another layer to our project, React Native can also make it more difficult to debug, particularly at the intersection of React and the host platform. Since React Native is still in its infancy, it presents the same challenges as those inherent in working with other novel technologies. Despite this, we will conclude, on balance, that the advantages exceed the disadvantages. To get a better understanding of what react-native is, let us first go through some of the restrictions of react-native:

- React Native is a Recently Developed Technology: React Native is still a new technology, and as we just stated, it has some drawbacks, kinks, and difficulties that need to be fixed. As a result, there is a Need to Fix These Issues. React Native is Still a New Technology 1. Because the framework needs specialized modules, developers may need more time to build and design brand-new modules from scratch. While evaluating the app, our partner company or the programmers must make this information known to us.
- The Need for Native Mobile Developers Because React Native's most favorable feature—implementing native code for faster performance—means that occasionally, React Native developers may require the assistance of native mobile app developers, it is essential to note that native mobile app developers are considered to be native mobile developers. The same principle applies to the application's AppStore and Google Play distribution. Native mobile developers often better understand the procedure involved in a successful launch and the supporting documentation.
- Designs that are Complicated and Their Interactions Not Compatible with the React Native Platform: The speed of React Native is pathetic when put in situations that require complex user interface design considerations, elaborate animations, and heavy interactions. This is also because of the concept of a bridge; all native modules are needed to link with the JavaScript component of the app, and if there are excessive interactions, the program may become unusable and sluggish as a result (Gutlapalli, 2017c).
- Dependence on Facebook: Using open-source technology from a third party, such as Facebook, comes with unique challenges. Other companies may follow Facebook's lead and discontinue using the technology due to their decision. Although Facebook uses React Native

in the main Facebook app, Instagram, Facebook Analytics, and the Ads Manager app, nothing suggests this new turn of events will occur. The neighborhood encompassing the framework continuously expands, and many big enterprises are involved in the conflict.

- Performance of Near-native Individuals Continues to Lag behind Performance of Native Individuals: Even though we have covered how the pace of React Native is superior to that of other cross-platform solutions, developing apps with it is still slower than native app development. It is recommended that we discuss the impact that React Native has on the performance of our app with the developers working on our team. It is a slight rise, hardly noticeable, for most apps, but it may become more prominent if it manages extensive data or "bridges" (Gutlapalli, 2016a).

COMPANIES USING REACT NATIVE

- Without Facebook, React Native's original author, the list of organizations using it would be incomplete. They sought to transfer web development's benefits—rapid iterations, a single team—to mobile. Facebook said they doubled start-up speed by changing one HTML element to React Native. The app would immediately benefit from updating to the latest React Native version because the framework improvements were applied directly.
- Instagram: React Native is utilized beyond Facebook's iOS and Android apps. Instagram started testing React Native in 2016, writing in a blog post that it speeds up both iOS and Android feature releases. They also released the percentage of code shared between applications to boost developer productivity.
- Walmart is an American multinational retailer that operates a chain of supermarkets. The app offers product search, in-store pricing comparison, and online shopping. They write 95% of their app code in React Native.
- Bloomberg: Bloomberg adopted React Native early in 2016 with their initial app release. Bloomberg app users get localized and time-of-day news. The business shift included plans to integrate React Native into other Bloomberg products.
- Tesla provides limited details on their decision to adopt React Native. All Tesla vehicles use the app as a trip companion. The software syncs with Android and iOS cellphones to control and monitor almost all Tesla functions, including the charger, panoramic roof, lights, locks, and horn.
- Shopify has identified React Native as its mobile future, intending to improve the app experience on iOS and Android platforms. They cooperated with Facebook and funded notable React Native community members to create a complete cross-platform mobile experience.

REACTJS/REACT NATIVE APPS

Companies that embraced ReactJS or React Native have done so for various reasons. The biggest adopters besides Facebook are listed below.

Instagram: Instagram wants to create a one-page app for online visitors to access the social platform. ReactJS seems best for this. Optimization made the web app fast and user-friendly. Instagram's mobile and web apps use React now.

Netflix: Netflix uses Gibbon, a rendering layer, with ReactJS, which it adopted in 2015. Netflix picked React for its declarative programming and one-way data flow concept.

Airbnb: The organization switched to ReactJS for its component reusability, simple code restructuring, and iterating. Internal structures of the company's mobile app and homepage utilize it.

UberEats: The platform was designed to connect restaurants, drivers, and clients. The trickiest task for developers was translating the restaurant dashboard from online to Android and iOS. React Native was the only option. Although only a portion of the Uber Eats mobile app is developed using React Native, the developers fulfilled their goals. The framework capability enabled service growth.

SoundCloud Pulse: When the SoundCloud team decided to create a mobile app for musicians to manage their accounts and copyrights, the music-sharing site had been around for ten years. Fitting into the three-person dev team and minimal resources was the biggest problem. React Native excelled for two reasons. The development process has been expedited since sharing the codebase. Second, developers benefited from RN's user-friendliness and automatic live code reloading.

Kahn Academy: Most of the largest online education platform uses ReactJS. Khan Academy used the framework to build its eLearning platform to avoid page reformation and replace elements quickly.

Bloomberg: For its mobile app, the online magazine used React Native. Their Tech at Bloomberg blog post calls it "the first tool that truly delivers on the promise of cross-platform native app development."

React's performance is impressive, considering how many firms use it. As other frameworks increase UI speed, the nascent and occasionally messy React Native wants to redefine how apps connect with software and hardware (Rodriguez, 2016; Gutlapalli, 2016b). However, ReactJS has grown from a simple debugging solution to a web development solution that optimizes and improves productivity.

How to Improve React Native App Performance

React Native is a potential solution for people seeking a feasible framework to construct cross-platform applications. It has been adopted worldwide by enormously successful firms such as Facebook, Shopify, Coinbase, Tesla, and Discord (Vipul et al., 2016). Why are some of the largest firms in the world shifting their app development to use React Native?

- It provides more concise and straightforward code to understand and can be shared across several platforms.
- It provides rapid iteration without the need for a compilation cycle.
- We will be able to deploy our product more quickly and concentrate on the most essential elements, thereby giving our app a beautiful appearance and feel.

However, optimization is an essential step that must be taken before our React Native app can reach its full performance potential. In this article, we will discuss specific optimizations that, when implemented, can make our React Native app substantially more efficient and run twice as fast. The amount of time it takes for an app to draw material from its database once launched is called the "start-up time." The start-up time can be improved by reducing the size of the bundle as well as the amount of RAM it uses. The app's performance can be improved by reducing the time needed to start.

Hermes is a free and open-source JavaScript engine explicitly tailored for RNs. Because activating Hermes will result in less memory use and smaller program size, we can use it to improve the time it takes for the application to start. When working with Herms, we should always use the most recent version of RN.

OPTIMIZE REACT CODE

The use of sound coding practices and methodologies can improve applications. As a result, it is recommended that we refrain from making render calls that are not essential and use anonymous functions whenever possible. These many render calls might lead to significant performance concerns if they must be appropriately managed (Potapov et al., 2018). We will not have to manually handle them if we utilize PureComponent to take care of them. This type of component does not modify the props or the state included within the element, preventing multiple render calls from being issued.

```
import React, {PureComponent} from 'react';

import { Text } from 'native-base';

classPureComponentExample extends PureComponent {

render() {

return<Text> This is a pure component</Text>;

 }

}

export default PureComponentExample;
```

Furthermore, react if we want to use the same concept in the function component. Memo has been introduced for the same purpose.

Use of memory optimization

We can use Xcode or Android Device Monitor to monitor any memory leaks in RN apps. Because of the large number of background processes that they contain, RN programs can occasionally cause memory leaks (Desamsetti & Mandapuram, 2017). As a result, we should stay away from these processes in the background as much as possible. For instance, if we wish to display a list view, it is advised that we use FlatList, SectionList, or VirtualizedList rather than ScrollView. This is because FlatList, SectionList, and VirtualizedList are more efficient. The application's efficiency is improved due to FlastList's implementation of lazy loading for the items.

FlatList example

```
import React from 'react';

import { SafeAreaView, View, FlatList, Text } from 'react-native';

const DATA = [

 {

id: '1',

title: 'First Item',
```

```
},
{
id: '2',
title: 'Second Item,'
},
{
id: '3',
title: 'Third Item',
},
];
const Item = ({ title }) => (
<View><Text>{title}</Text></View>
);
const App = () => {
constrenderItem = ({ item }) => (
<Item title={item.title} />
);
return (
<SafeAreaView><FlatList
data={DATA}
renderItem={renderItem}
keyExtractor={item => item.id}
    /></SafeAreaView>
);
}
export default App;
```

Use of memory optimization

When debugging JavaScript code, using console logs is a widespread practice. However, keeping those console statements even after the application has been deployed would result in significant performance concerns owing to the JavaScript thread (Lal et al., 2018). Therefore, we may eliminate any console statements from the application using a Babel plugin.

The following instructions will allow us to install the babel plugin successfully.

npm install babel-plugin-transform-remove-console

```
# OR
yarn add babel-plugin-transform-remove-console
```

Then, we can modify the .babelrc file to remove console statements from the production env, as shown below.

```
{
  "env": {
    "production": {
      "plugins": ["transform-remove-console"]
    }
  }
}
```

Make Use of Uncontrolled Inputs

When a user types very quickly, the controller input often causes the device to slow down and can cause rendering issues when updating the view. This is the most prevalent problem with controller input (Thaduri et al., 2016). Additionally, because it must also deal with native JavaScript threads, controller input is noticeably more sluggish than uncontrolled input. On the other hand, uncontrolled inputs in RNs have a higher performance because modifying them does not result in a state change.

```
export default function UncontrolledInputs() {
const [text, onTextChange] = React.useState('Controlled inputs');
return (
<TextInput
style={{ height: 100, borderColor: 'red' }}
onChangeText={text =>onTextChange(text)}
defaultValue={text}
    />
  );
}
```

A React Native application's performance may be negatively impacted by a variety of reasons, including the presence of huge images, intensive computations, and superfluous render calls (Desamsetti, 2016). Following best practices and utilizing tools such as those presented and recommended in this article will help us prevent many of the frequent performance difficulties that arise.

CONCLUSION

The fact that the application is native while still using standard web technologies, like JavaScript, is the most appealing aspect of working with React Native. In other words,

applications designed with React Native run as quickly and smoothly as any native application built with classic Android and iOS technologies such as Objective-C, Swift, or Kotlin.

An exciting new technology called React Native allows web developers to construct robust mobile applications utilizing their existing knowledge of JavaScript by leveraging this framework. It enables quicker mobile development and code exchange that is more effective across iOS, Android, and the Web, all without compromising the quality of the experience provided to the end user or the applications themselves. The disadvantage is that it's brand new and there is still a lot of room for improvement. We should have a look at React Native to see if our team can withstand the unpredictability that comes with working with new technology and wants to develop mobile applications for more than just one platform.

React Native applications are also similar to native applications built with these old technologies. On the other hand, unlike other popular hybrid frameworks that rely on web technologies to construct iOS and Android applications, React Native does not make any sacrifices in terms of performance and the overall experience. In addition to this, this is the reason why, ever since its release, React Native has become a framework that is generally acknowledged for the building of mobile applications. Learn once, write anywhere is an approach that may be perfectly justified when applied to the React Native framework because there is no requirement to learn a fundamentally different set of technologies for each mobile platform.

REFERENCES

Aggarwal, S. (2018). Modern Web Development using ReactJS. *International Journal of Recent Research Aspects, 5*(1), 133-137. http://ijrra.net/Vol5issue1/IJRRA-05-01-27.pdf

Bodepudi, A., Reddy, M., Gutlapalli, S. S., & Mandapuram, M. (2019). Voice Recognition Systems in the Cloud Networks: Has It Reached Its Full Potential?. *Asian Journal of Applied Science and Engineering*, *8*(1), 51–60. https://doi.org/10.18034/ajase.v8i1.12

Dekkati, S., & Thaduri, U. R. (2017). Innovative Method for the Prediction of Software Defects Based on Class Imbalance Datasets. *Technology & Management Review*, *2*, 1–5. https://upright.pub/index.php/tmr/article/view/78

Deming, C., Dekkati, S., & Desamsetti, H. (2018). Exploratory Data Analysis and Visualization for Business Analytics. *Asian Journal of Applied Science and Engineering*, *7*(1), 93–100. https://doi.org/10.18034/ajase.v7i1.53

Desamsetti, H. (2016). Issues with the Cloud Computing Technology. *International Research Journal of Engineering and Technology (IRJET), 3*(5), 321-323.

Desamsetti, H., & Mandapuram, M. (2017). A Review of Meta-Model Designed for the Model-Based Testing Technique. *Engineering International*, *5*(2), 107–110. https://doi.org/10.18034/ei.v5i2.661

Fedosejev, A., Prusty, N., Horton, A., Vice, R., Holmes, E., & Bray, T. (2016). *React: Building modern web applications*. Packt Publishing, Limited.

Gutlapalli, S. S. (2016a). An Examination of Nanotechnology's Role as an Integral Part of Electronics. *ABC Research Alert*, *4*(3), 21–27. https://doi.org/10.18034/ra.v4i3.651

Gutlapalli, S. S. (2016b). Commercial Applications of Blockchain and Distributed Ledger Technology. *Engineering International*, *4*(2), 89–94. https://doi.org/10.18034/ei.v4i2.653

Gutlapalli, S. S. (2017a). Analysis of Multimodal Data Using Deep Learning and Machine Learning. *Asian Journal of Humanity, Art and Literature*, *4*(2), 171–176. https://doi.org/10.18034/ajhal.v4i2.658

Gutlapalli, S. S. (2017b). The Role of Deep Learning in the Fourth Industrial Revolution: A Digital Transformation Approach. *Asian Accounting and Auditing Advancement*, *8*(1), 52–56. https://4ajournal.com/article/view/77

Gutlapalli, S. S. (2017c). An Early Cautionary Scan of the Security Risks of the Internet of Things. *Asian Journal of Applied Science and Engineering*, *6*, 163–168. Retrieved from https://ajase.net/article/view/14

Hitz, B. C., Rowe, L. D., Podduturi, N. R., Glick, D. I., Baymuradov, U. K., Malladi, V. S., Chan, E. T., Davidson, J. M., Gabdank, I., Narayana, A. K., Onate, K. C., Hilton, J., Ho, M. C., Lee, B. T., Miyasato, S. R., Dreszer, T. R., Sloan, C. A., Strattan, J. S., Tanaka, F. Y., . . . Cherry, J. M. (2017). SnoVault and encodeD: A novel object-based storage system and applications to ENCODE metadata. *PLoS One*, *12*(4). https://doi.org/10.1371/journal.pone.0175310

Kowalczyk, K., & Plechawska-Wójcik, M. (2016). AngularJS and ReactJS libraries - performance analysis. *Journal of Computer Sciences Institute*, *2*, 114-119. https://doi.org/10.35784/jcsi.126

Krill, P. (2015). Facebook extends React Native to Android. *InfoWorld.Com*, https://www.proquest.com/trade-journals/facebook-extends-react-native-android/docview/1712288805/se-2

Lal, K., Ballamudi, V. K. R., & Thaduri, U. R. (2018). Exploiting the Potential of Artificial Intelligence in Decision Support Systems. *ABC Journal of Advanced Research*, *7*(2), 131-138. https://doi.org/10.18034/abcjar.v7i2.695

Mandapuram, M. (2016). Applications of Blockchain and Distributed Ledger Technology (DLT) in Commercial Settings. *Asian Accounting and Auditing Advancement*, *7*(1), 50–57. https://4ajournal.com/article/view/76

Mandapuram, M. (2017a). Application of Artificial Intelligence in Contemporary Business: An Analysis for Content Management System Optimization. *Asian Business Review*, *7*(3), 117–122. https://doi.org/10.18034/abr.v7i3.650

Mandapuram, M. (2017b). Security Risk Analysis of the Internet of Things: An Early Cautionary Scan. *ABC Research Alert*, *5*(3), 49–55. https://doi.org/10.18034/ra.v5i3.650

Mandapuram, M., & Hosen, M. F. (2018). The Object-Oriented Database Management System versus the Relational Database Management System: A Comparison. *Global Disclosure of Economics and Business*, *7*(2), 89–96. https://doi.org/10.18034/gdeb.v7i2.657

Mandapuram, M., Gutlapalli, S. S., Bodepudi, A., & Reddy, M. (2018). Investigating the Prospects of Generative Artificial Intelligence. *Asian Journal of Humanity, Art and Literature*, *5*(2), 167–174. https://doi.org/10.18034/ajhal.v5i2.659

Paul, A., Nalwaya, A. (2016). The Simplest Program: Hello World with React Native. In: React Native for iOS Development. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-1395-7_2

Pinto, C. M. and Coutinho, C. (2018). From Native to Cross-platform Hybrid Development. 2018 International Conference on Intelligent Systems (IS), Funchal, Portugal. 669-676, https://doi.org/10.1109/IS.2018.8710545

Potapov V. P., Popov S. E., Kostylev M. A. (2018). Information and computing system for massively parallel processing of radar data in the Apache Spark environment. *Computational Technologies, 23*(4), 110-123. http://www.ict.nsc.ru/jct/annotation/1863

Rodriguez, J. (2016). Microsoft acquisition opens the door for cross-platform mobile application development. *Cio*, https://www.proquest.com/trade-journals/microsoft-acquisition-opens-door-cross-platform/docview/1768285144/se-2

Sengupta, D., Singhal, M., & Corvalan, D. (2016). *Getting started with react: A light but powerful way to build dynamic real-time applications using reactjs*. Packt Publishing, Limited.

Thaduri, U. R., Ballamudi, V. K. R., Dekkati, S., & Mandapuram, M. (2016). Making the Cloud Adoption Decisions: Gaining Advantages from Taking an Integrated Approach. *International Journal of Reciprocal Symmetry and Theoretical Physics*, *3*, 11–16. https://upright.pub/index.php/ijrstp/article/view/77

Thodupunori, S. R., & Gutlapalli, S. S. (2018). Overview of LeOra Software: A Statistical Tool for Decision Makers. *Technology & Management Review, 3*(1), 7–11.

Vipul, A. M., Sonpatki, P., & Sonpatki, V. A. M. P. (2016). *Reactjs by example - building modern web applications with react: Get up and running with reactjs by developing five cutting-edge and responsive projects*. Packt Publishing, Limited.

--0--