International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 5, Issue 1, ISSN No. 2455-2143, Pages 579-585
Published Online May 2020 in IJEAST (http://www.ijeast.com)

IMPLEMENTING REACT TO PROVIDE PARKING SOLUTIONS

Kartikey Dwivedi
Department. of Computer Science & Engineering
Presidency University,
Bengaluru, Karnataka, India

Dr KG Mohan
Department of Computer Science & Engineering
Presidency University
Bengaluru, Karnataka, India

***Abstract*— Building web applications to provide a solution for a problem has always been the best choice because it saves the developer time instead of creating two different applications for iOS and Android. Recently, there have been so many technologies introduced which drastically changed our thought about web applications. "Park@work" is a web-based application the solves the common problem of parking in offices, malls, and hospitals in an effective and simple way. It provides an inexpensive solution to this enormous problem of parking.**

This project explains why we should use react over other libraries to build websites and also how we can build a dynamic, reusable component using React effectively, and why it's far efficient in terms of performance and lines of code when compared to other technologies used for building websites.

***Keywords*—React JS, React hooks, material design, lazy loading, Suspense, SaaS model.**

I. INTRODUCTION

According to the survey of Netcraft's January 2020 survey, there are over 1.7 billion websites on the world wide web and this number keep changing every second. The websites play a very important role in building the nation's economy whether it is for trade, business such as E-commerce or social media websites like Facebook, Twitter or Instagram.

As per the survey of Netcraft's, there is around 2 zettabytes of data available on the internet and it keeps changing every second. Hence to handle this huge amount of data every company upgrades their website with the latest technologies to make it efficient so that it can easily handle the traffic which is increasing day by day. To overcome the issue of performance and efficiency in 2011 Facebook engineer Jordon Walke and his team introduced the world with a new technology named React. React is a JavaScript library for building web interfaces managed by Facebook and a community of developers and other companies. React was first used by Facebook in their Facebook Newsfeed in 2011. Since than React community has been kept flourishing with new developers getting added every day. Currently many successful companies like Facebook, Dropbox, Tesla, Airbnb, Netflix, Reddit, etc. has implemented react in building their website. Facebook uses over 30,000 react components to manage their website, because of React Facebook has been able to handle a huge portion of the market in the world.

As per the survey of business time the parking industry market is worth over 400 billion dollars hence it's extremely necessary for the parking companies to build the applications using cutting edge technologies like the react library. But currently in Indian parking sector, no company has come with a website that works on react, hooks, and material UI.

"Park@work" is a system that has integrated react and other new technologies to provide parking solutions effectively. The proposed work has two major components- A parking dashboard and Web view link. The parking dashboard has been developed using react and material design which manages the parking at the admin end. And, the web view link which allows users to do hassle-free parking.

To operate the parking dashboard, the admin will be provided with the login credentials through which they can access the data of the customer provided by the customer in the web view link during sign-up. The dashboard has a dynamic table component that is designed using material design, where admin can perform different CRUD operations. Whenever they add, edit, or delete the data, the table gets refreshed automatically. The table is also provided with a sorting, search, and export data feature to provide ease to the user to handle data. Through the unique pin provided to the customer, they can log in to the web view link and can choose parking space in their office building.

The main reason to build this website is to show how does react reduces the amount of work, developers have to do, and how it overcome a problem which is faced by almost all parking buildings. Currently, in India, no parking application manages the parking of offices digitally. One of the problems which every parking office face is the amount of time they spend on parking their vehicle. With the implementation of the Park@work project, this issue can be resolved effectively.

II. LITERATURE SURVEY

Currently, in India, there are so many parking applications like best parking, parker, spot Hero which provides parking solutions. They aim to provide parking to any location like on-street, malls, hospitals, etc. They have a predefined solution which they manage and integrate it into the operational site. But providing such solutions is very expensive as it required a camera, sensors and sometimes the customer has to use their mobile apps for parking. There are few other solutions which are given to resolve the problem of parking.

Robin Grodi et al. (2016) [2] built a network of sensors using an induction proximity sensor, UV sensor, and RFID sensors. They used these sensors to detect the availability of a parking spots in the parking lot. The network of sensors is handled using Arduino Uno. To provide their solution they built a mobile application which shows the availability of parking space.

Shen-En Shih and Wen-Hsiang Tsai et al. (2014) [3] used wide-angle camera attached to the ceiling which detects the parking boundary and analyze the free parking space through Hough transform with the new cell accumulation scheme.

Manickam, Sunil and Elango et al. (2018) [4] implemented IoT to provide parking solution. They used a series of ultrasonic sensor and raspberry pi which coordinate together and display the name of free parking lots. They used a third-party app BLYNK which is connected with the ESP-8266 wirelessly which gives the information of free parking space to user.

Ajay and Surekha et al. (2018) [5] applied Dijkstra's algorithm to find the nearest possible parking space. They used cloud computing to store the user data and manage the parking space through Arduino and other sensors.

Although these technologies are efficient to a certain level, they fail to provide an inexpensive solution. Park@work resolves this problem by providing a dashboard through which the admin can share a unique pin, to the user through SMS and email service [9], and the customer can use the web link to do parking. With the help of that unique pin the end-user can sign-in in the weblink portal and can view the current status of parking spots left in a building through the status and progress bar of each building.

When the customer leaves the building, they will also exit from the portal and that spot will be shown available again. With the help of the dashboard, the admin can not only access the real-time data but can also perform operations like add user, edit user details, delete user data and copy their record in case the admin wants to share the details with the customer. These features will be necessary if this solution is used in any office, hospital, or other places where they have employees doing parking every day. The admin can add their details like vehicle no, contact no and email. After successful data record creation, a message will be sent to the employee on their contact no which will contain a unique pin and the WebView link through which they can access the status of parking space available in a building. Since usually a company handles more than one building the employee can see the parking spaces left in a building and can go for parking in the next building.

In the dashboard the admin has been provided with a search field where they can search for any respective record. Apart from that, there is also be a sort feature that will allow the admin to sort the data in increasing or decreasing order. If the admin wants to get the entire data in an excel format then there is an export data feature that downloads an excel sheet with all the data present in the table. Since the project is built using React hooks, there are very few lines of code written to design this website. Moreover, using react has improved the increase in the efficiency of the website by decreasing the page reloading time.

III. BENEFITS OF REACT HOOKS

There are so many reasons why Park@work project is build using react. Below mentioned points explain why react is more advanced than any other library.

1. It facilitates the overall process of writing applications of any kind because it accepts the HTML quoting and makes a subcomponent rendering easier. Besides that, it provides informative warning and error message and it also prevents code injections.
2. It enhances productivity and also facilitates further maintenance because it uses reusable components which are wrapped under other components and are managed through the root component. While building the project there were many buttons, navbar, sidebar, footer, or other small components that were same for all the pages. With the help of react library the same code is rendered in all pages to display the components.
3. It ensures faster rendering as it uses the virtual document object model instead of a real document object model. Virtual document object model is basically a virtual representation of user interface which is stored in memory and synced with the real document object model by a library such as React document object model. This process is called reconciliation.
4. It ensures stable code as ReactJS use only downward data flow. Hence changing an object requires only the modification of its state and only that component will be updated. This structure of data binding ensures

code stability and improves performance.

5. Using react hooks encapsulate side effects and improve the readability of component tree.

6. It is SEO friendly because it runs on the server hence it renders and returns the virtual DOM to the browser as a regular webpage.

IV. SYSTEM ARCHITECTURE

A. *THREE- TIER ARCHITECTURE*

The system architecture that Park@work follows is 3-tier architecture as depicted in Fig. 1.

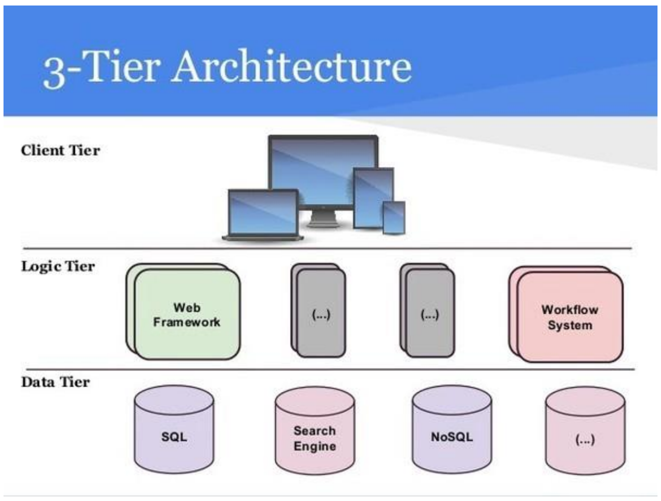


Figure 1: Three- tier architecture

1) *Client Tier:* In this tier, the actual discussion is purely on the user interface design and quality attributes that were used as part of designing the user interface. This tier is the only gateway where any user can communicate with the product model and gets their jobs done, so the user interface which the users are expected to use should be simple and minimalist. All the components build are reusable and scalable so in later stages modification is easy to implement.

2) *Logic Tier:* This tier mainly includes the algorithms, different frameworks that were used while handling data from backhand server. The flow of data fetch is as follows,
Client request from UI → Data processing in Application Tier → Call API → request hit to backhand server → Query processing in server→ Fetch data from database → response hit back to Application tier → Select data from the response received → response shown to user in view tier.

So, in above-shown process of pulling and pushing of data into the backhand server, there is a need of an accurate framework that can process the data and render it in web view. We have used frontend libraries like React, Material design. With the help of React Hooks, props, and class component the API was successfully called from the backhand. Different hooks such as useEffect and useState handles the current state of the component and whenever we call any API the data reloads through Lifecycle components.

3) Data Tier: In Park@work whenever the user sends the request from frontend, the get API is called, through which the instruction is given to the backhand which internally calls the database and send the data back with the response, the data comes from the backhand user through post API.

B. *WORKFLOW OF PARK@WORK WEBSITE*

This section gives the overview of the components which are utilized by Park@work website.

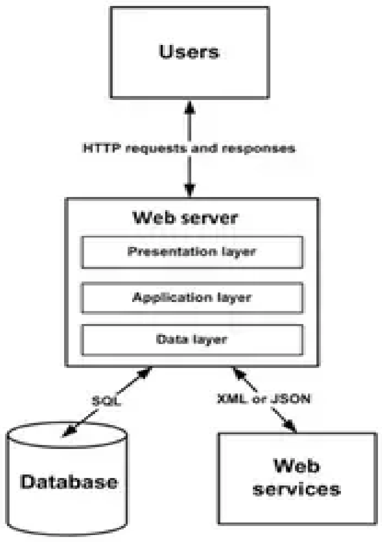


Figure 2: Work Flow of website

1.) *Users:* Makes requests to the webserver and receive responses using PHP Server pages.

2.) *Web server:* Hosts the application's various layers which conform with the reusable component-
Presentation Layer: Users interact with the application via HTTP requests and responses renders in the browser.
Application Layer: Manage the flow of the

application, implements business logic and liaises with the data layer to process the request from users and their responses.
Data Layer: Handles domain data and provides persistence and retrieval services for the database.

3.) *Database:* Where data is persisted and retrieved. Park@work data is stored in a MongoDB database. It handles the data efficiently and provides security to it.

4.) *Web services:* Interaction with other applications. To interact with the webserver the Park@work dashboard and weblink use RESTful APIs (it uses HTTP request to GET, POST, PUT and DELETE data). To manage such requests Axios and Http Utils are used which are the package of react library. Through them, request is sent to the backhand and if the status is true than response is received which stores the information requested by the user.

C. FLOW DIAGRAM OF THE PROJECT

The flowchart explains the flow which application will follow while being operated by the user.

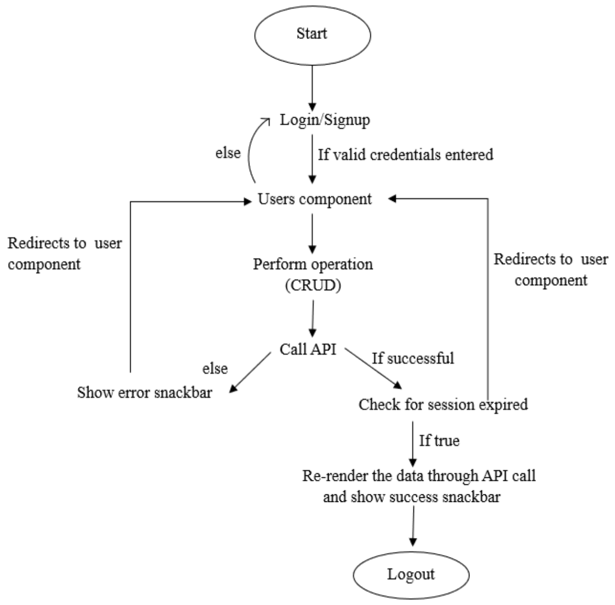

Start
Login/Signup
else
If valid credentials entered
Users component
Redirects to user component
Perform operation (CRUD)
Redirects to user component
Call API
else
If successful
Show error snackbar
Check for session expired
If true
Re-render the data through API call and show success snackbar
Logout

Figure 3: Flow diagram of the project

The diagram explains the different scenario which user will face and how the system will respond to each use-case. The diagram is simplified and minimalized and gives an overview of the dashboard. It starts from the login screen and explain the different scenario and use case and ends with the logout screen. In between, if the user enters an invalid URL in any screen then the user will be redirected to the "Page not found" component.

On successful login the user is provided with a specific token key and in case if any other user tries to login in the website the previous user session will get expired and the user will be notified with a snack bar message of "Session expired" and will be redirected to the login page. Park@work project has also used material design to improve the user experience. Material design is an android-oriented design language created by Google that provides rich features, built-in component and natural motions that mimic real-world objects. Material design has its own set of components like button, table, grid, forms, etc. It helps to customize the website easily and make it elegant with its custom features.

The description of each component is mentioned below:

1) *Login Component*- It will be the first webpage when the user loads the URL. It consists of a top navbar and an email and password field. If the user will enter the correct email and password then the user will be redirected to the user component. Otherwise, in case of incorrect data the color of the fields with get changed to red and will show an error message. Apart from that, the sign-in button will be disabled.

2) *Users Component*- On successful sign-in, the user is redirected to the user's component which is basically a dashboard. This page contains a topbar, footer, sidebar, and a body. Except for the main body, every other component is a reusable component and it is rendered for every other page without writing any new code again. This is one of the pros of why the project is build using react. As such parking website contains a lot of reusable components and with the help of react, we can easily render even a button into a new page without writing any new line of code. React allows to use any component just by wrapping in inside other component and manage it through the root component.

3) *Table Component*- It's the most important component of this project as it holds the data of all the users who have parked their vehicles in the building. The table component loads in the main body of Users component. Since it's a dynamic table it provides many features as discussed below:

Add user: It allows the admin to create a new record of an employee who wants to use the parking area. The table consists of five fields as vehicleno1, vehicleno2, vehicleno3, phone, and email. To create a record its mandatory to the enter vehicle number in the vehicleno1 field. This feature helps the admin to save the data of their customer early so that later he can easily park his vehicle by entering the pin provided by the admin through a message.

Edit user: This feature provides the admin to edit any data in the table. This helps the admin to modify the present record instead of creating a new record for the user. After editing the data an API will be called and if it passes all the validation rules which are set for all the fields than the success message will pop-up on the screen and table with be reloaded again.

Delete user: In case the admin wants to delete any user record than they can click of delete icon and a pop-up dialog box will appear asking if the admin really wants to delete the record. If the user clicks on the CANCEL button then the website will reload the previous state. On click of OK button, the data get deleted and the user is shown with a success message and the table gets reloaded again.

Copy data: If the admin wants to copy any particular row data, it can be done by clicking on copy data icon and the data gets copied to the clipboard and a success message is shown on the screen as “copied”.

Share pin: In case the customer forgets his unique pin then they can request the admin to share the pin. The Park@work website has a share pin icon on click of which a dialog box will appear asking if the admin really wants to share the pin to the contact no mentioned. If clicked yes, an API will be called and the customer will receive a message on his mobile with a unique pin and a WebView link.

Search data: In case if admin wants to search any data in the table, they can just enter that text in the search field and if it matches with the data present in the table than that data rows will appear in the table. If no data matches with the data entered in the search field than a “No data found” message is displayed on the message.

Sort data: With this feature, the admin can sort any column data in increasing or decreasing order. It helps the admin to view the data as per their choice. Upon sorting the page is set back to 0 and the user can navigate to the next page through table pagination.

4) *Logout Component*- It allows the admin to sign-out from the dashboard and redirects to the sign-in page. It clears the session and deletes the token. Hence, in case if the user tries to redirect back to the User component the page reloads back to the sign-in page.

5) *Not Found Component*- This component is written in case if the admin enters some invalid URL in between hence it will redirect the user to this component and will request the user to navigate back to the previous page.

V. RESULTS AND DISCUSSION

A. Performance view

In order to enhance the performance of the project, lazy loading concept was implemented [1]. Lazy loading is a function in react that load react components lazily through code splitting without taking any help from any additional libraries. Lazy-loading renders only-critical interface items first, then quietly unrolls the non-critical items later. The code has been built using react.lazy() method to improves the performance of the project. This lazy function has been wrapped around suspense component which takes a fallback property that accepts the react elements we would like to render as the lazy component is being loaded.

With the help of lazy loading, the performance of the project was increased several times. In order to measure the performance of the website the browser built in profiler was used. Chrome browser provides real-time data which depicts how much time and bytes it takes to render data from the server. It records the time of loading files from the server. It also stores other information like memory used, API calls, and network history. All these data in represented in graphs and charts which can be analyzed easily.

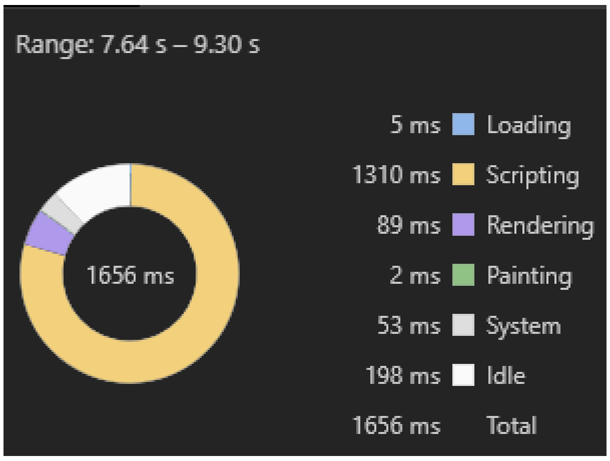
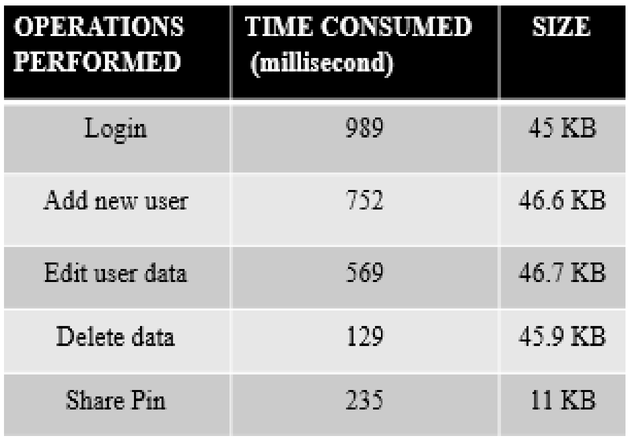


Figure 4: Website rendering time from server side

To measure the performance of the website the performance measuring feature was enabled in the browser that records the time from initial request made by the user to the page rendering in front-end.

Figure 4 depicts that to render the entire Park@work website it took around 1.6 seconds whereas usually, it takes on an average 7-8 seconds to load the entire website. The performance profiler was also used to collect the real-time data for all API calls to the backhand. The major observations that were inferred from the performance profiler were the overall analysis of API calls time and size of data they return to the front-end.

The below-mentioned table depicts the performance of Park@work website during network calls.



OPERATIONS PERFORMED	TIME CONSUMED (millisecond)	SIZE
Login	989	45 KB
Add new user	752	46.6 KB
Edit user data	569	46.7 KB
Delete data	129	45.9 KB
Share Pin	235	11 KB

Figure 5: Analysis of Park@work dashboard

B. *SaaS Model*

Since this project was designed keeping SaaS model in mind it is easily scalable. The project automatically customizes the theme of the website based on the response from the backhand. The API response is checked on successful login, and based on the company name the theme and logo on the website get updated. Such features allow this project to be used by multiple organizations without writing new code.

VI. CONCLUSION

The goal of this project is to develop an inexpensive solution which can be implemented and operated easily in offices, hospitals and malls. This project effectively solves the problem which people faces in their offices or the places where they visit regularly. As this project is built keeping SaaS model in mind, it is easily scalable and can be used by all. Since this project uses React and material design it explains why all website should incorporate such library in their projects. With the help of such technologies the performance of the website and web link has been increased by 80%. Especially parking website which uses other technologies like Java and Angular JS, can use this technology and can build some advanced website which works phenomenal in terms of performance and efficiency.

ACKNOWLEDGMENT

The author would like to acknowledge the support of our guide Dr. K G Mohan and we would also like to acknowledge numerous discussions on the subject with him.

VII. REFERENCES

[1] Tuaycharoen, N., Prodpran, V., & Srithong, B. (2018). ClassSchedule: A web-based application for school class scheduling with real-time lazy loading. 2018 5th International Conference on Business and Industrial Research (ICBIR). doi:10.1109/icbir.2018.8391194

[2] Grodi, R., Rawat, D. B., & Rios-Gutierrez, F. (2016). Smart parking: Parking occupancy monitoring and visualization system for smart cities. SoutheastCon 2016. doi:10.1109/secon.2016.7506721

[3] Shih, S.-E., & Tsai, W.-H. (2014). A Convenient Vision-Based System for Automatic Detection of Parking Spaces in Indoor Parking Lots Using Wide-Angle Cameras. IEEE Transactions on Vehicular Technology, 63(6), 2521–2532. doi:10.1109/tvt.2013.2297331

[4] Ramasamy, M., Solanki, S. G., Natarajan, E., & Keat, T. M. (2018). IoT Based Smart Parking System for Large Parking Lot. 2018 IEEE 4th International Symposium in Robotics and Manufacturing Automation (ROMA) doi:10.1109/roma46407.2018.8986731

[5] Zajam, A., & Dholay, S. (2018). Detecting Efficient Parking Space Using Smart Parking. 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT). doi:10.1109/icccnt.2018.8493964

[6] Y. Geng and C. G. Cassandras, "A new smart parking system based on optimal resource allocation and reservations," in Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on. IEEE, 2011, pp. 979–984.

[7] Parikh, S. P., Shah, D. R., & Shah, P. R. (2018). Park Indicator: Book Your Parking Spot. 2018 Fourth International Conference on Computing Communication Control and Automation (ICCIBEA). doi:10.1109/iccubea.2018.8697378

[8] D. Vakula and Y. K. Kolli, "Low cost smart parking system for smart cities," 2017 International Conference on Intelligent Sustainable Systems (ICISS), Palladam, 2017, pp. 280-284.

[9] N. H. H. M. Hanif, M. H. Badiozaman, and H. Daud, "Smart parking reservation system using short message services (SMS)," in Intelligent and Advanced Systems (ICIAS), 2010 International Conference on, 2010, pp. 1–5.

[10] A. Ghannem, M. S. Hamdi, M. Soui, and H. Ammar, "An adaptive iparking application: An ontology-based approach," in Future Technologies Conference (FTC). IEEE, 2016, pp. 777–78

[11] Sophie Alpert and Dan Abramov.(2018) Introduction to React hooks https://reactjs.org/docs/hooks-intro.html

[12] Google Team (2018) Material UI documentation https://material-ui.com/getting-started/installation/

[13] Jordan Walke (2013) React Documentation https://reactjs.org/docs/getting-started.html