ISSN: 2455 - 1597

ijergs
Open Access

International Journal of Engineering Research and Generic Science (IJERGS)
Available online at: https://www.ijergs.in

Volume - 11, Issue - 2, March – April - 2025, Page No. 01 - 09

Developing Progressive Web Apps (PWA) with the MERN Stack

[1]Sagar Pradhan, Assistant Professor, Department of Computer Science, Arya College of Engineering, Jaipur.

[2]Shivang Baranwal, Research Scholar, Department of Computer Science, Arya College of Engineering, Jaipur

[3]Utkarsh Kumar Singh, Research Scholar, Department of Computer Science, Arya College of Engineering, Jaipur.

Abstract

Progressive Web Applications (PWAs) have emerged as a transformative technology that bridges the gap between traditional web applications and native mobile apps. They leverage modern web capabilities to deliver app-like experiences to users directly through their browsers, without requiring installation from app stores. Key features such as offline functionality, background synchronization, push notifications, fast loading times, and the ability to be installed on home screens have made PWAs an increasingly popular choice for developers aiming to enhance user engagement and reach broader audiences.

Simultaneously, the MERN stack—comprising MongoDB, Express.js, React.js, and Node.js—has gained widespread adoption due to its simplicity, efficiency, and use of a single programming language (JavaScript) across the entire development stack. This full-stack framework enables rapid development and seamless integration of frontend and backend components, making it a powerful tool for building scalable and maintainable web applications.

Integrating PWA capabilities within MERN-based applications offers a unified and robust approach to developing high-performance, cross-platform solutions. This paper delves into the foundational concepts of PWAs, examines how each component of the MERN stack contributes to the development lifecycle, and presents a detailed methodology for building PWAs using the MERN stack. The study includes real-world case studies and architectural diagrams to illustrate the synergy between these technologies. Furthermore, it explores the opportunities and benefits for developers and businesses, addresses the common challenges faced during development and deployment, and highlights future prospects of PWA adoption in the industry. Through this exploration, the paper aims to provide a comprehensive guide for developers and researchers interested in harnessing the power of modern web technologies

Keywords: Progressive Web Apps, MERN Stack, MongoDB, Express.js, React.js, Node.js, Web Development, Full-stack JavaScript, Service Workers, Offline-first Applications.

Introduction

In recent years, user expectations for web experiences have significantly evolved, driven by the rapid adoption of smartphones, high-speed internet, and a demand for more seamless interactions with digital platforms. Users now expect web applications to perform nearly as fast as native apps, with near-instant loading times, responsive interactions, offline functionality, and smooth transitions between pages. A great deal of this shift is due to the increasing importance of Progressive Web Apps (PWAs), which aim to meet these expectations and bridge the gap between the functionality of native applications and the accessibility of web-based solutions.

Progressive Web Apps leverage modern web technologies to provide a user experience that mimics the behavior of native apps while being hosted and accessed through a web browser. PWAs are equipped with key features such as service workers, which enable the app to work offline or with a limited network connection by caching important resources and data. This ensures that users can still interact with the app even when they are not connected to the internet, eliminating one of the most common limitations of traditional web apps. Additionally, PWAs utilize a web manifest file, which allows the app to be installed on a user's device and accessed like a native application, often through a home screen icon, and with minimal loading times, ensuring a smoother user experience.

In tandem with this evolution, the MERN stack has become an increasingly popular toolset for developing modern, scalable web applications. The MERN stack consists of MongoDB, Express.js, React.js, and Node.js, each playing a specific role in enabling developers to create robust applications quickly and efficiently. One of the most significant advantages of the MERN stack is its use of JavaScript across the entire development stack, from the client side (React.js) to the server side (Node.js and Express.js), and the database layer (MongoDB). This uniformity allows developers to work in a single language throughout the development process, facilitating faster development, better code reusability, and easier collaboration between front-end and back-end teams.

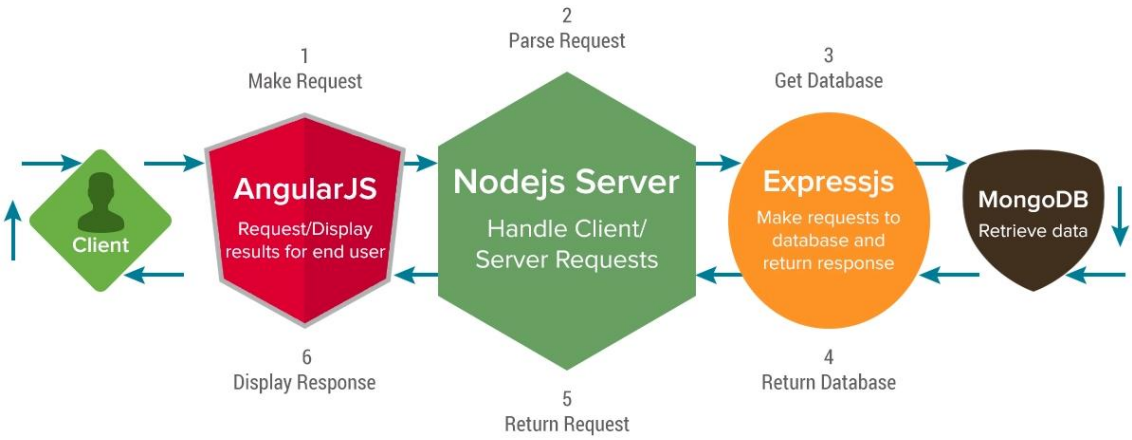
React.js, a key component of the MERN stack, is a powerful JavaScript library for building user interfaces. Its component-based architecture allows for reusable UI elements, making it highly suitable for creating dynamic, interactive, and user-friendly applications. React can also be combined with tools like Create React App, which provides a convenient setup for building single-page applications with PWA capabilities. By integrating service workers into React, developers can ensure that their applications are fully capable of offline functionality, which is a core feature of PWAs.

Express.js, another important tool in the MERN stack, acts as the server-side framework that provides routing, middleware, and API management. It simplifies handling HTTP requests and responses, making it easier to interact with databases, manage sessions, and process user data. Node.js serves as the runtime environment for Express, enabling JavaScript code to be executed server-side. Together, Express.js and Node.js form the back end of the application, enabling the handling of API requests, server logic, and real-time features, such as notifications or live updates.

MongoDB, the NoSQL database in the MERN stack, plays a critical role in managing data for modern web applications. Its document-based structure allows for flexible and scalable data storage, which is ideal for applications that handle large amounts of unstructured data. MongoDB integrates seamlessly with the other MERN stack technologies, providing a consistent experience for developers and ensuring that data is easily accessible and can be updated dynamically.

When the capabilities of PWAs are integrated with the MERN stack, the result is a highly performant application that offers seamless offline support, installability on devices, and an overall improved user experience. For example, developers can use React (with Create React App) to enable service workers and configure a manifest file, ensuring that resources are cached and the app can function even without a network connection. Express.js and Node.js handle the server-side logic and API management, while MongoDB efficiently handles the storage and retrieval of data.

This powerful combination of PWA features and the MERN stack allows developers to create scalable, high-performance web applications that meet the expectations of modern users. Whether it's an e-commerce platform, a social media site, or a productivity tool, integrating PWA capabilities into MERN-powered applications ensures faster load times, offline accessibility, and an overall native-like experience for users, which ultimately leads to higher user satisfaction and engagement.



Recent examples

1. A notable example of a MERN-PWA implementation is a Task Manager app developed by a student project team, which showcases the seamless integration of both the MERN stack and PWA capabilities. This application was designed to allow users to create, update, and delete tasks while offering full offline support. The unique advantage of this system was its ability to work effectively even when users were not connected to the internet. When online, the app synchronized with a backend Node.js server, which was connected to MongoDB Atlas, ensuring that all tasks were updated and stored properly in the cloud.

2. The service worker, a core feature of the Progressive Web App, played a vital role in enabling offline functionality. It cached essential assets and API calls, ensuring that users could continue managing their tasks even without an active internet connection. This caching mechanism not only allowed the app to function in offline mode but also significantly improved load times when users were reconnected to the internet, delivering a smoother and faster experience. In addition to this, the app was equipped with a manifest.json file, which allowed users to install the app on both mobile and desktop devices. By acting similarly to a native app, users could access the Task Manager directly from their home screens, with no need to open a web browser. This installation capability made the app feel like a native mobile or desktop application, further enhancing user experience and increasing engagement.

3. Another compelling example is the development of an e-commerce platform using the MERN-PWA combination. This platform was designed to provide customers with an enhanced shopping experience, including the ability to browse and purchase products even when offline. The platform relied heavily on the PWA's offline capabilities to ensure that product information and images were available to customers regardless of their connection status. By utilizing service workers,

the app cached product data and images, so users could continue their shopping experience without disruption, even if they temporarily lost access to the internet.

4. Moreover, the integration of push notifications added a significant layer of interactivity to the platform. Customers received timely alerts about new product offers, flash sales, or order updates, keeping them engaged and informed at all times. This feature, which was made possible through service workers and the Push API, was a key element in enhancing the overall user experience, providing value and convenience even when the user was not actively browsing the site.

5. From a business perspective, the adoption of the PWA model led to a remarkable 30% increase in the conversion rate. Customers were more likely to complete their purchases due to the

6. consistent and reliable performance of the platform, regardless of their internet connection. The ability to browse and make purchases offline reduced the friction often experienced by users in traditional e-commerce sites that required constant connectivity. Additionally, businesses observed improved engagement, with customers returning to the app more frequently, as the PWA model offered a faster, more reliable, and native-like experience compared to traditional web applications.

7. By leveraging the MERN stack for backend development and utilizing PWA capabilities for front-end performance enhancements, both of these projects demonstrate the powerful synergy that can be achieved. The combination of React.js for dynamic user interfaces, Express.js and Node.js for scalable backend APIs, and MongoDB for flexible data storage provides a solid foundation for building high-performance applications. When enhanced with service workers, caching, and offline capabilities, these apps not only meet the increasing demands of users for faster, more reliable experiences but also enable businesses to drive better results in terms of user retention, engagement, and conversion rates.

8. These examples underline the immense potential of MERN-PWA applications in various sectors, whether it's task management, e-commerce, or any other domain. The ability to deliver a seamless, fast, and interactive user experience across different devices and network conditions has made PWA a game-changer for modern web development, and the MERN stack has proven to be an excellent choice for building these kinds of applications

Opportunities and Benefits

The integration of Progressive Web Apps (PWAs) with the MERN stack brings a wealth of exciting opportunities and benefits for both developers and businesses. As modern web development continues to evolve, the ability to combine the scalability, flexibility, and performance of the MERN stack with the seamless, native-like experience offered by PWAs positions businesses to meet the growing expectations of users while ensuring robust and efficient development processes. Below are some of the key advantages that stand out when integrating PWAs with the MERN stack:

1. Cross-platform Compatibility

One of the most significant benefits of PWAs is their ability to work seamlessly across a variety of devices and operating systems. Unlike traditional mobile apps, which require separate development for each platform (iOS, Android, etc.), PWAs are built using standard web technologies—HTML, CSS, and JavaScript—ensuring that they function consistently on any device with a browser. This includes desktops, laptops, smartphones, and tablets, regardless of whether the user is on Windows, macOS, Android, or iOS. For developers, this cross-platform capability eliminates the need to develop and

maintain separate codebases for different platforms, which streamlines development and reduces the time and resources spent on platform-specific optimizations.

2. Improved Performance

Performance is a critical factor in ensuring a positive user experience, and PWAs shine in this area. Service workers, a key feature of PWAs, cache essential assets, including images, CSS files, and JavaScript, allowing them to be served locally rather than from the server. This reduces server load and minimizes latency, speeding up the app's responsiveness. Cached assets not only contribute to faster load times but also ensure that the application can work in situations with limited or no internet connectivity. This means that users can continue using the app without interruption, whether they are on a slow network or temporarily offline. As a result, the user experience becomes smoother, which is a significant factor in retaining users and improving engagement.

3. Offline Support

Offline support is another compelling benefit offered by the integration of PWAs with the MERN stack. By using service workers, PWAs can cache data and resources, enabling users to access and interact with the app even when they do not have an internet connection. This is especially important in areas with unreliable network access, or for users who may be on the move, such as travelers or those with limited connectivity. Offline functionality ensures that users can still perform essential tasks, such as viewing content or interacting with their personal data, even without an active internet connection. Once the user reconnects, the app synchronizes with the backend server, ensuring data consistency across devices and networks.

4. Reduced Development Costs

PWAs offer significant cost-saving opportunities for developers and businesses alike. Since a single codebase is used for both mobile and desktop users, the development and maintenance costs are considerably lower compared to traditional app development, which requires separate codebases for different platforms. With the MERN stack, the backend and frontend can be developed using a consistent language—JavaScript—which further streamlines the development process. This consistency between the client-side (React.js) and server-side (Node.js, Express.js) not only makes development more efficient but also simplifies debugging and testing. As a result, businesses can reduce the time spent on app development, focus resources on enhancing core features, and bring their product to market faster.

5. Increased User Engagement

The integration of features such as push notifications and app install prompts can significantly enhance user engagement. Push notifications allow businesses to send timely updates, reminders, and personalized offers directly to users, even when the app is not open. This feature is particularly useful for e-commerce platforms, social media applications, or news sites that rely on user interaction. The ability to prompt users to install the PWA on their devices further increases engagement, as it makes the app more accessible and convenient. When users install the app, they can launch it directly from their device’s home screen, creating a more native-like experience and improving overall retention rates.

6. SEO Friendliness

Unlike traditional native apps, PWAs are web-based and can be indexed by search engines. This is a significant advantage for businesses looking to drive organic traffic to their web applications. Search engine optimization (SEO) is essential for improving visibility and attracting users, and PWAs offer the same SEO benefits as regular websites. By using standard HTML, CSS, and JavaScript, PWAs can be crawled, indexed, and ranked by search engines, allowing businesses to reach a broader audience without relying solely on app store listings. This also allows for content to be easily discovered by users searching for relevant keywords, further boosting the app's visibility and user acquisition.

7. Better User Retention and Conversion Rates With features like offline support, fast loading times, and a native-like experience, PWAs can significantly improve user retention. Users are more likely to return to an app that works smoothly, offers consistent performance, and provides access to content and features even when offline. As mentioned in the e-commerce example, businesses that implement PWAs have seen marked improvements in conversion rates, as the application provides a smoother, more reliable shopping experience. By providing features that increase convenience—such as offline browsing, push notifications, and fast load times—businesses can foster higher engagement and conversion rates, ultimately leading to greater revenue.

8. Lower Bandwidth Usage

PWAs can reduce bandwidth usage by caching key resources and serving them locally. This is particularly beneficial for users with limited data plans or those in regions with slower internet speeds. By minimizing the number of server requests and serving cached resources, PWAs allow users to consume less data, which can help improve their experience, particularly in areas where internet connectivity is a concern. This not only benefits users but can also reduce server load, improving scalability and reducing operational costs for businesses.

Challenges

Despite its many strengths, the integration of Progressive Web Apps (PWAs) with the MERN stack does come with a set of technical and operational challenges that developers need to address to ensure the success of the application. These challenges arise from both the inherent complexity of web development and specific limitations of the technologies involved. Understanding and mitigating these challenges is crucial for developers to fully leverage the potential of MERN-PWA applications. The following are some of the key challenges developers may face:

1. Service Worker Complexity

Service workers are at the heart of PWAs, enabling offline functionality, background sync, and caching. However, they also introduce a level of complexity, particularly when dealing with advanced caching strategies. Managing cache effectively to ensure that users always get the most up-to-date resources without overloading the server is a non-trivial task. Developers need to define strategies for cache invalidation, network-first or cache-first approaches, and fallback mechanisms for when resources are unavailable. Improper cache management can lead to outdated content being served, poor user experiences, and synchronization issues when users go back online. The complexity increases when scaling the app to handle different types of resources (e.g., static files, dynamic data) and ensuring that data consistency is maintained across sessions and devices.

2. iOS Limitations

While PWAs offer a largely consistent experience across devices and platforms, there are some significant limitations when it comes to iOS devices. For example, iOS Safari does not fully support all the features of PWAs, which can hinder the overall user experience on iPhones and iPads. One notable limitation is that iOS does not allow service workers to function as effectively as on other platforms, leading to issues with caching and offline functionality. Push notifications, a crucial feature of many PWAs, are also not supported on iOS. This means that businesses targeting iOS users may not be able to take full advantage of all the capabilities that PWAs offer, potentially leading to a less unified experience across devices. Although Apple has started to improve support for PWAs, some features, such as full-screen experience and app-like behavior, are still inconsistent compared to Android or desktop platforms.

3. Initial Load Time

React applications, especially large and complex ones, can experience high initial load times if they are not optimized properly. React apps often require significant resources for their initial rendering, including downloading JavaScript files, stylesheets, and other assets. If these resources are not efficiently managed or bundled, it can result in a poor first-time load experience. Slow load times not only frustrate users but can also affect search engine rankings, user retention, and conversion rates. Developers must implement techniques such as lazy loading, code splitting, and tree shaking to minimize the size of the initial JavaScript bundle and improve load times. These optimizations are essential, particularly when integrating a PWA with React, as a smooth and fast first-time user experience is crucial for gaining and retaining users.

4. SEO Limitations

Search engine optimization (SEO) is a critical aspect of any web application, especially for businesses looking to increase organic traffic and visibility. However, React apps, by default, present a challenge for SEO. React uses a client-side rendering (CSR) model, which means the content of the page is rendered by JavaScript in the browser rather than on the server. Search engines often struggle to index JavaScript-heavy websites, which can negatively impact the app's search engine ranking and discoverability. This issue can be mitigated through server-side rendering (SSR), where the initial HTML is generated on the server and then sent to the browser. However, implementing SSR with React requires additional configuration and resources, and not all hosting environments are conducive to SSR. For PWAs built with the MERN stack, ensuring that the app is both SEO-friendly and capable of leveraging SSR may require extra development effort and infrastructure.

5. Storage Management

Service workers enable offline functionality by caching assets and data locally, but browsers impose storage limits on the amount of data that can be stored for offline use. These limits vary depending on the browser and device, and in some cases, can be restrictive for applications that need to store large amounts of data. For instance, an e-commerce platform with many products

images or a task management app with a large amount of user-generated data may encounter storage issues if the data exceeds the available cache storage. Developers need to carefully manage storage to ensure that the application remains functional without exceeding these limits.

This may involve using techniques like cache prioritization, limiting the number of items stored, or periodically cleaning up the cache to free up space. Striking a balance between providing offline functionality and managing storage constraints can be a complex task, especially as the application scales.

6. Security Risks

While PWAs offer many benefits, they also introduce potential security risks if not properly implemented. Service workers, while powerful, can expose vulnerabilities if they are not coded correctly. A poorly configured service worker can allow malicious scripts to intercept requests or serve outdated or harmful content to users. Additionally, if service workers are not adequately secured, they can be a vector for man-in-the-middle attacks, compromising user data. It is crucial to ensure that all communication between the client and server is done over HTTPS, and service workers are registered and configured securely. Developers should also be cautious about cache management, ensuring that sensitive data is never stored in the service worker cache. Regular security audits, proper input validation, and using tools like Content Security Policy (CSP) headers can help mitigate security risks.

References

1. Gautam, S. (2020). Developing a Pet Finder PWA using MERN Stack.
2. Lama, N. (2019). Providing native experiences in mobile with PWA. US/docs/Web/API/Service_Worker_API
3. Love, C. (2018). Progressive Web Application Development by Example: Develop fast, reliable, and engaging user experiences for the web. Packt Publishing Ltd.
4. Hoque, S. (2020). Full-Stack React Projects: Learn MERN stack development by building modern web apps using MongoDB, Express, React, and Node. js. Packt Publishing Ltd.
5. Nguyen, B. (2021). Improving web development process of MERN stack.
6. Alamin, M. (2022). A social platform for software developers: Using modern web stack MERN.
7. Martola, H. (2021). An evaluation of implementing a progressive web application with micro frontend architecture.
8. 8 Yberg, V. (2018). Native-like performance and user experience with Progressive Web Apps.
9. Yener, Y. Progressive Web Apps-Costs, Benefits and Tradeoffs.
10. Wahlström, M. (2017). Exploring progressive web applications for health care: Developing a PWA to gather patients' self assessments.
11. Johannsen, F. (2018). Progressive Web Applications and Code Complexity: An analysis of the added complexity of making a web application progressive.
12. Baral, P. (2020). Role-based User Access Control in MERN Stack applications.13. Ionic Blog. PWA Adoption Cases. https://ionic.io/resources/articles
13. 13.VENKATRAMAN, V. Project Report: Interactive Web Content for STEM.15. Medium. Building Offline-First React Apps. https://medium.com/.

14. Yadav, C., Dhakad, R., Panchal, M., & Kaur, E. B. (2024). Revolutionizing Near-by Accommodation: An in-depth Analysis of React. js, Node. js, MongoDB, and Express. js Integration for Website
15. Development. Bhupinder, Revolutionizing Near-by Accommodation: An in-depth Analysis of React. js, Node. js, MongoDB, and Express. js Integration for Website Development (August 21, 2024).
16. Sumathy, G., Maheshwari, A., AR, A., Sherin Shibi, C., & Kanimozhi, N. (2024, December). StudentsConnect: A Web Application to Track Library Permissions and Leave Requests in Hostels Using MERN. In 2024 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES) (pp. 1-8). IEEE.
17. Sumathy, G., Maheshwari, A., AR, A., Sherin Shibi, C., & Kanimozhi, N. (2024, December). StudentsConnect: A Web Application to Track Library Permissions and Leave Requests in Hostels Using MERN. In 2024 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES) (pp. 1-8). IEEE.
18. GIROLAMI, M. (2021). Sviluppo di una full stack chat app utilizzando lo stack MERN.
19. Gatla, S., Santosh Krishna, B., Anitha, M., & Rajendra Prasad, K. (2024, March). Cloud Storage Platform Using the MERN Stack. In International Conference on Innovations in Cybersecurity and Data Science Proceedings of ICICDS (pp. 421-435). Singapore: Springer Nature Singapore.
20. Petr, P. (2022). Moderní technologie a přístupy k vývoji webových aplikací (Master's thesis, České vysoké učení technické v Praze. Vypočetní a informační centrum.).
21. Oliveira, L. R. M. (2019). Productizing a Mobile Application: Participatory Design and Development (Master's thesis, Universidade de Aveiro (Portugal)).