NextGen Scientific Publication
American International Journal of Computer Science and Technology

Volume 7 Issue 3, Pg. No. 137-147, AIJCST-V7I3P111, 2025
https://doi.org/10.63282/3117-5481/AIJCST-V7I3P111

*Original Article*

# Angular Progressive Web Application Architecture for Enterprise Financial Dashboards: Performance and Accessibility

***Hari Krishna Mupparapu[1], Chandra K Movva[2]**

[1]*Senior .NET Developer, GM Financial, Charlotte, NC, USA.*
[2]*Senior Android Developer, Bass Pro Shops & Cabela's, Springfield, MO, USA.*

## Abstract:

*Enterprise financial dashboards present demanding requirements for Progressive Web Applications built with Angular, combining large dataset visualization, real-time data refresh, role-differentiated views, and strict accessibility compliance for regulated financial environments. This paper examines Angular PWA architectural patterns optimized for enterprise financial dashboard delivery.*

## Keywords:

*Angular, Progressive Web Applications, Financial Dashboards, Enterprise Frontend, Performance Optimization, Accessibility, WCAG, Service Workers, Lazy Loading, Web Performance.*

## Article History:

***Received: 04.04.2025***

***Revised: 08.05.2025***

***Accepted: 18.05.2025***

***Published: 29.05.2025***

## 1. Introduction

The increasing digital transformation of the financial sector has accelerated the adoption of web-based enterprise dashboards for monitoring transactions, portfolio performance, risk indicators, and operational metrics. Frontend applications are vital in today's financial landscape, offering real-time data and critical features to meet the security, responsiveness, and accessibility standards of modern financial institutions. [1] Progressive Web Applications (PWAs) have come to the rescue by offering the access of web technologies and the security and experience of native apps. Angular is one of the few frameworks available that provides a solid foundation for building robust, scalable, and maintainable enterprise-level PWAs with its modular architecture, built-in routing, dependency injection, and strong TypeScript support.

Enterprise financial dashboards present unique architecture challenges because they have to process massive amounts of data, continuously sync to other data streams, and provide varying views based on user roles and permissions. Moreover, financial institutions in regulated environments need to adhere to accessibility guidelines like Web Content Accessibility Guidelines (WCAG) to ensure that all the functionalities of the dashboard are accessible to various user groups. Thus, the use of performance optimization strategies, such as service workers, efficient state management, and lazy loading, is crucial for ensuring smooth performance and consistent user experience.

The purpose of this paper is to examine architectural patterns and best practices for the creation of enterprise financial dashboards that are built using Angular-based Progressive Web Apps. It explores ways to make applications perform better, make applications work better off-line, and to add accessibility to application development process. The study showcases the potential of today's Angular PWA architectures to power scalable, secure, and accessible financial platforms that can keep up with the changing needs of enterprise users.

## 2. Literature Review

### 2.1. Enterprise Dashboard Architectures

Enterprise dashboard architectures have undergone substantial transformation in response to the growing demand for real-time business intelligence and data-driven decision making. [2] Financial dashboards are used in financial environments to create centralized interfaces that collect, manipulate and display complex operational and transactional information and provide executives and analysts a view of the key performance indicators (KPIs). One of the main goals of these systems is to deliver information in a concise but interactive way, to improve the cognitive understanding. The principles of dashboard usability that have been most studied include high data-ink ratio, minimal visual clutter, contextual filtering, and drill down, which overall helps to minimize information overload and improve user understanding.

Modern enterprise dashboards are usually designed with several layers of technical architecture, such as presentation, application, data integration, and infrastructure layers. Architectural frameworks have identified a number of design principles and evaluation criteria to support the design of a scalable and maintainable dashboard ecosystem. Another benefit of modular architectures is the ability to develop and deploy individual dashboard features without affecting the coupling between functional modules and maintain dashboard features over time. By using widgets that can be reused and user interfaces that can be customized, organizations can enhance dashboard functionality without impacting the current workflow or necessitating significant architectural overhauls.

Architectural considerations go beyond visualization and modularity in enterprise financial dashboards, to real-time analytics, role-based personalization and constant data synchronization. Financial analysts want to be able to tailor the layout of their dashboard, create dynamic reports and receive alerts automatically when certain thresholds are reached or anomalies are identified. Therefore, to ensure dashboard performance even during high traffic loads, it is important to pair streaming data pipelines with effective front-end rendering strategies. Time to Interactive (TTI), memory usage and smooth frame rendering are now critical performance metrics for dashboard quality. This has spurred the introduction of frontend architectures that can manage vast amounts of data and still provide a seamless and easy-to-navigate user experience.

### 2.2. Angular Framework for Enterprise Systems

Angular has established itself as one of the leading frameworks for enterprise-scale web application development due to its structured architecture, strong tooling ecosystem, and built-in support for large development teams. Angular offers a suite of tools, from routing to dependency injection to forms to test utilities to internationalization and the state management, unlike lightweight frontend libraries. [3] This opinionated design philosophy promotes uniformity among projects and help organizations build and maintain large-scale applications while ensuring that all their coding is in standard mode and thus mitigating technical debt. The framework's TypeScript-first design also promotes software quality by performing static type checking, leading to more reliable and maintainable code for mission-critical enterprise software.

Financial applications are a key use case for Angular, as it can be used to create secure, scalable, and highly interactive applications. The component-driven development of Angular encourages the use of reusability, which means that many of the interface components commonly used in large organizations, like charts, transaction tables, filters, and reporting modules can be built once and reused across the entire application. This modularity can speed up development cycles and bring uniformity in the user experience. Further, Angular's dependency injection architecture makes it easier to test and maintain applications, making direct application module dependencies minimal.

Angular's performance and reactivity has greatly improved with recent updates, including the arrival of standalone components and Signals. Signals help update the state of a component at a finer level of granularity, which helps to avoid unnecessary re-rendering of components and also helps to improve the responsiveness of data intensive interfaces. When coupled with optimized change detection strategies like OnPush, an Angular application can efficiently process and display thousands of financial records, without degrading the user experience. Plus, its built-in HTTP client, RxJS observables, and WebSocket support ensure seamless integration with RESTful APIs, message brokers, and real-time data streams. The features make Angular well suited for enterprise financial dashboards that always need to be updated, act deterministically and perform well under demanding operational conditions.

### 2.3. Progressive Web Application Technologies

A Progressive Web Applications (PWA) is a fully grown and viable way to deliver enterprise software that sits somewhere between a typical web app and the native mobile app. [4] PWAs offer the following capabilities in addition to being accessible via traditional web browsers: Offline access, background synchronization, installability, push notifications, etc., using modern web standards. The pair allows companies to provide the kind of rich app-like experience without the cost and complexity of running multiple native applications. PWA technologies are supported in most modern browsers and the mobile browser capabilities have been enhanced significantly, which has further propelled their adoption in enterprise settings.

The foundation of PWA architecture consists of several core technologies. Service workers are background workers that can work offline and cache content, as well as run background sync operations, to keep applications running even when network conditions are poor. The web app manifest provides information about the way a web application will be installed, the icons and themes that will be used, and the preferences that will be set at launch time so that the application can be installed directly from the browser onto the device. Together with secure HTTPS communication and responsive design principles these technologies make very reliable and platform independent applications that can be used in the enterprise. The native support allows developers to add enterprise-class offline support with minimal configuration overhead, while optimizing application loading performance. Financial dashboard scenarios are especially beneficial with PWA features that enable users to view the cached reports, navigate through the application modules, and still work when the connection is down for a while. Efficient application shell architectures and smart caching strategies also ensure faster load times and lower bandwidth usage, thereby enhancing user experience.

Organized on an economic and operational level, PWA is an interesting solution. One codebase for both desktop and mobile platforms means PWA solutions save in development and maintenance costs compared to native application development. Significant cost savings have been reported with function comparable to native applications. In addition, PWAs make services easily accessible by eliminating the need to download an application, and by making the service immediately available in standard web browsers. With generative accessibility that includes assistive technologies, and adherence to accessibility standards like WCAG, PWA is an appealing architectural option for enterprise financial dashboards that need to be high-performing, accessed by many and easy to use.

## 3. Requirements Analysis and System Design

### 3.1. Functional Requirements

The scope of functional needs for an enterprise financial dashboard developed on the basis of Angular PWA architecture is concerned with providing the business intelligence accurate, real-time and relevant to a specific role. The system needs to be secure, allowing access of user authentication and role based access control, so that administrators, financial analysts, managers and auditors have access to only information and operations that are relevant to their specific job responsibilities. [5] Its features encompass real-time data visualization via interactive charts and tables, customizable dashboards, advanced search and filtering capabilities, report generation and connectivity with external financial data sources via REST API and WebSocket services. Automated alerts and notifications should also be included in the application for any major events like market fluctuations, abnormal transactions, and threshold breaches. There are also certain functional features that are key for offline viewing, such as the caching of updated information through the use of service workers and enabling the synchronization of updated information when Internet connectivity is restored, as well as responsiveness across desktop and mobile devices.

### 3.2. Non-Functional Requirements

The non-functional requirements are the quality attributes needed to support financial dashboard reliability, scalability, and usability in the enterprise. Performance is essential, with pages loading quickly, Time to Interactive (TTI) being minimal, and a very responsive interface for users even when loading large amounts of data. The system should be scalable to accommodate growing numbers of concurrent users and volumes of financial data, without impacting system performance. [6] Security standards involve using HTTPS for encrypted communication, implementing safe authentication procedures, safeguarding against common web vulnerabilities, and following financial data protection policies. The accessibility should meet WCAG criteria, i.e., the interface should be navigable by keyboard, compatible with screen readers, offer adequate color contrast and be adaptable to various abilities. Additionally, maintainability and extensibility are crucial, and should involve a modular design of the Angular architecture that supports reusability, ease of testing and future feature additions that will not disrupt current operations.

### 3.3. Enterprise Financial Dashboard Use Cases

The financial dashboard is an excellent tool for enterprise financial management and can be utilized in various activities across financial institutions and corporate organizations, including; strategic decision making, operational management, financial monitoring, and analysis. [7] For example, financial analysts might use customizable widgets and dynamic charts on dashboards to track real-time market trends, portfolio performance, and transaction histories. Managers and executives rely on aggregated KPI dashboards to gain a real-time view of organizational performance, to assess risks and to take business decisions in time based on live financial information. Different role-specific interfaces help compliance officers and auditors to review reports from regulations, suspicious activities, and compliance with internal and external governance processes. The dashboard also enables multi-user workflows: they can export reports, share visualizations, and even get automated notification alerts once a predefined business rule or risk threshold is met. The system's real-time analytics, secure access control and offline PWA feature allows for continuous financial operations even with intermittent network connectivity.

## 4. Proposed Angular PWA Architecture

### 4.1. Overall System Architecture

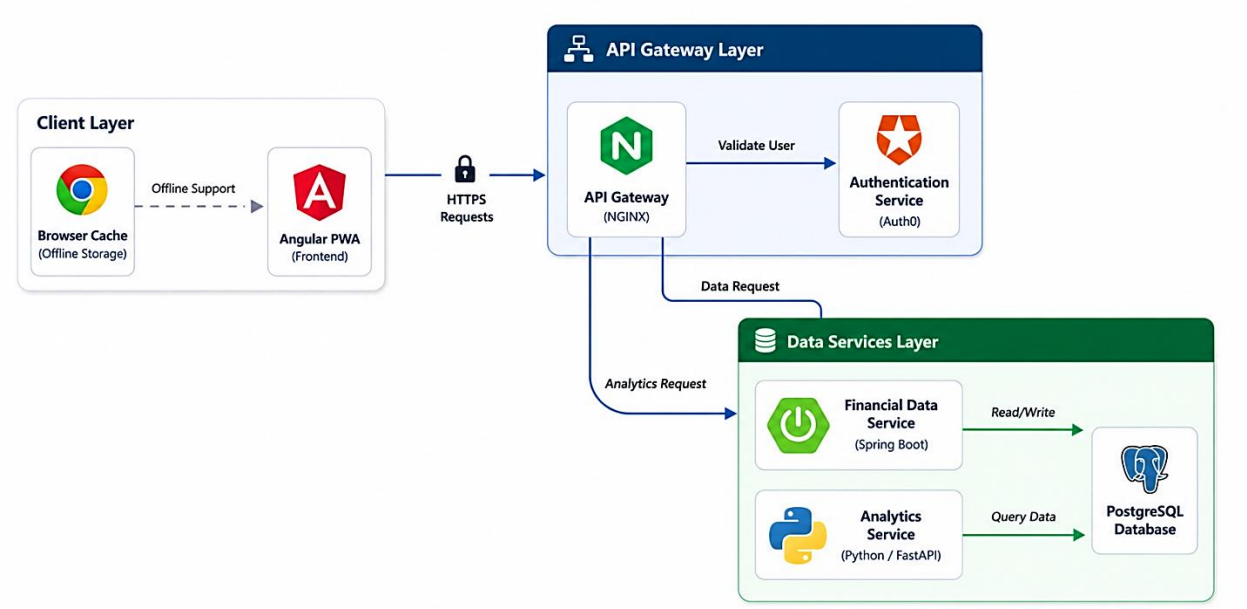


**Figure 1. Proposed Angular Progressive Web Application Architecture for Enterprise Financial Dashboards**

The proposed Angular Progressive Web Application architecture adopts a layered design to provide scalability, security, and high performance for enterprise financial dashboard environments. The Angular PWA is the main user interface at the client side, and the use of browser cache, service worker technologies, etc. provides offline experience and speeds up the delivery of content. [8] The dashboard is accessed by users securely over HTTPS and there is encrypted communication between the frontend and backend infrastructure. It's designed to provide a responsive user experience, leveraging client-side rendering and intelligent caching techniques to minimize latency and enhance the availability of the applications, even in the event of fluctuating internet access.

The back end infrastructure is structured around an API Gateway that routes and validates requests, and guards access control. The API gateway uses NGINX to validate the user requests with the authentication service and then passes on the authorized requests to the data services layer. In this layer, there is a financial data service that performs transactions and business logic, and another analytics service that processes analytical workloads and advanced data queries. They both communicate with a PostgreSQL database for storing/accessing financial data. This modular design encourages loose coupling of components which allows for scaling components independently, maintenance easiness and easy integration of more enterprise services. The proposed architecture is robust and scalable, ensuring that the financial dashboard applications are secure, accessible, and resilient.

### 4.2. Angular Application Architecture

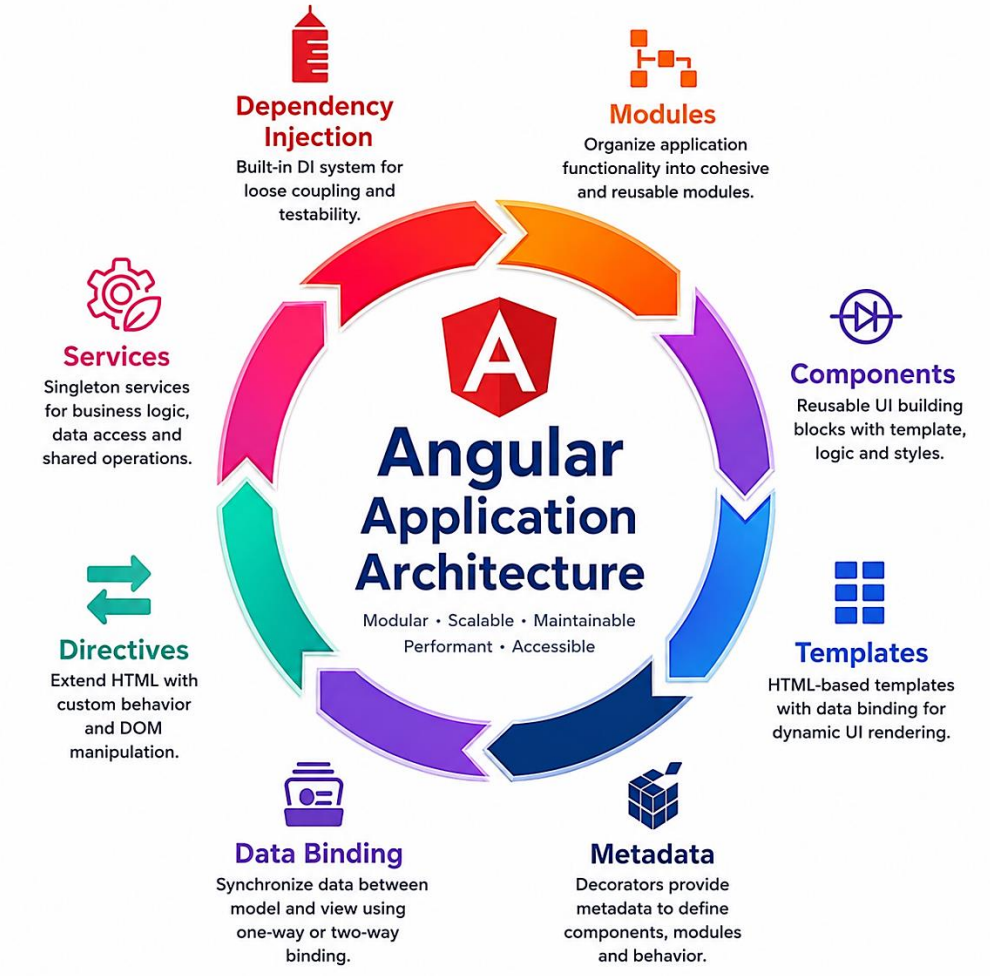


**Figure 2. Core Angular Application Architecture for Enterprise Financial Dashboard Development**

The Angular application architecture shown in Figure 2 illustrates the modular and component-driven design philosophy that makes Angular highly suitable for enterprise-scale Progressive Web Applications. The core of the architecture is the Angular application itself, while there are several interconnected building blocks that act as a solid foundation for the creation of maintainable and scalable systems. [9] The modules group together related functionality in reusable units, and the components group together user interface logic and presentation, which can be used to build interactive dashboard components like charts, transaction tables, and analysis widgets. Templates and data binding mechanisms make it easy to propagate changes in financial data to the user interface, allowing for up-to-the-minute dashboard updates without complicated coding.

Separation of concerns, services, dependency injection also play a role in the architecture, as do isolating business logic and data access from presentation layers. Services offer shared functionality like API communication, authentication, and data processing, and Angular's dependency injection framework makes it easier to test and maintain the application and reduces tight coupling. Directives are used to add HTML behavior to make it one that offers dynamic user interactions, while metadata decorators are used to specify the configuration and relationships among the application elements. These architectural elements can be combined to create modular high-performing and accessible dashboards for enterprise financial data that will effectively handle large volumes of data, be updated in real-time, adjust to changing business needs, and be long-term maintainable and high-quality.

### 4.3. Progressive Web Application Layer

The Progressive Web Application (PWA) layer further enriches the Angular-based Enterprise Financial Dashboard by adding native application-like capabilities using modern web technologies. [10] This layer will use service workers, browser caching and the web app manifest to achieve fast loading times, offline capabilities and smooth installation on both desktop and mobile. Service workers store app elements and financial reports that are accessed often, and users can still see important information if the network connection is temporarily lost. Background synchronization mechanisms guarantee that locally stored data is updated automatically when it is connected again. The PWA layer has been built to combine responsive design with an efficient application shell, which enhance the users' experience, lower bandwidth usage and ensure access to financial dashboards in various operational contexts.

### 4.4. Financial Data Integration Layer

The financial data integration layer serves as a medium of communication between the front-end application and information sources in the enterprise, and it allows information to be exchanged in real-time and securely. It's connected to external market data providers, financial management systems, transactional databases and analytical services via RESTful APIs, WebSocket connections, and microservice interfaces. Data aggregation and data transformation mechanisms are used to standardize the data received from multiple heterogeneous sources, before displaying it inside the Angular dashboard. The architecture allows for low-latency responses and consistency in data, enabling seamless integration of financial metrics, portfolio updates, and key performance indicators, while also facilitating continuous streaming. The service integration model allows the system to be easily scaled up to handle larger amounts of data and new enterprise services, while maintaining optimal performance on the front end.

### 4.5. Security and Authentication Layer

The security and authentication layer provide the necessary security and compliance to enable enterprise financial dashboards to run in a legally governed environment. [11] Centralized identity providers and secure authentication services handle user authentication, using the OAuth 2.0 and OpenID Connect protocols for identity verification and session management. Role-based access control (RBAC) mechanisms enforce authorization policies: Users are only able to make use of data and functions that are appropriate to their roles within the organization. All communication between the AngularPWA and any of the backend services are secure via HTTPS encryption, and API gateways can be an extra layer of security that validates the requests and controls access to the services. Together with secure token management, input validation, and ongoing monitoring, this multi-layer safety approach secures delicate monetary data without compromising on consumer-friendliness or experience.

## 5. Performance Optimization Framework

### 5.1. Front-End Performance Strategies

For enterprise financial dashboards, front-end performance optimization is crucial, as users need to quickly access real-time data and interact with visualizations. Several strategies are used for the proposed Angular PWA architecture to reduce initial bundle sizes, page load times and application responsiveness such as lazy loading feature modules, code splitting, etc. Angular components and standalone modules can be used multiple times, reducing code duplication and making them easier to maintain, and with RxJS observables for asynchronously loading data, the interfaces don't block while communicating with the backend. Furthermore, responsive designs and optimized asset delivery provide a consistent user experience on all devices, such as tablets, mobile devices and desktops, so that the dashboard is usable in all scenarios, network speed and conditions.

### 5.2. Rendering Optimization

Rendering optimization aims at minimizing the amount of unnecessary updates to the user interface, and at having a smooth visualization of a large set of financial data. It leverages efficient change detection methods of Angular like OnPush, and Signals-based state management, to update only the components that were affected by data changes, rather than re-rendering the whole application. [12] Virtual scrolling and paging are used to display thousands of records in large transaction tables and analytical reports without using too much memory. Additionally, optimized chart rendering and incremental data updates boost frame rates, fostering smooth interactions and ensuring that real-time dashboard components remain responsive, even when handling high data volumes.Moreover, optimized chart rendering and incremental data updates enhance frame rates, resulting in smooth interactions and ensuring that real-time dashboard elements remain responsive even during intensive data processing.

### 5.3. Caching and Resource Management

Caching and resource management are fundamental to the reliability and efficiency of Progressive Web Applications in enterprise financial environments. The new architecture uses service workers to store the application shell, static content, and resources that are used often, and will allow the app to load quickly even when offline. The intelligent caching strategies differ between static and dynamic content, ensuring that the most important financial data is updated regularly and keeping redundant network requests to a minimum. Local cache storage and background synchronization keep locally stored information up to date, even if there is no connection, and provide access to critical dashboard capabilities. These methods, along with optimized memory management, asset minification, and image compression, help to minimize bandwidth and enhance the scalability and user experience of any application.

## 6. Accessibility Engineering Framework

### 6.1. Accessibility Architecture

Figure 3 illustrates the proposed accessibility architecture integrated into the Angular Progressive Web Application framework for enterprise financial dashboards. The architecture is layered, starting with the user layer where there are a variety of ways to interact with the device: by typing, using screen reader, or using a mobile device. A dedicated accessibility layer is integrated into these interaction modes, with Web Content Accessibility Guidelines (WCAG) compliance, [13] keyboard navigation controls and Accessible Rich Internet Applications (ARIA) standards. It acts as an intermediate layer between the interface and the underlying platform to ensure that interface elements are meaningful, fully accessible without a mouse and supported by assistive technology, which allows for access by users with varying needs.

The accessibility layer is closely coupled to the Angular PWA layer, with the creation of reusable accessible components and responsive user interface elements to provide a cohesive user experience across a variety of devices and platforms. The architecture additionally comprises a validation layer that is accountable for automatic accessibility tests and conformity checks. The validation process continually checks accessibility rules, focus management, semantic labeling and responsive behavior, creating compliance reports to assist developers in tackling accessibility problems during the software lifecycle. This framework, featuring user-centric design principles and automated validation, also provides a solid basis for creating enterprise financial dashboards that are efficient, standards-based, and user-friendly.

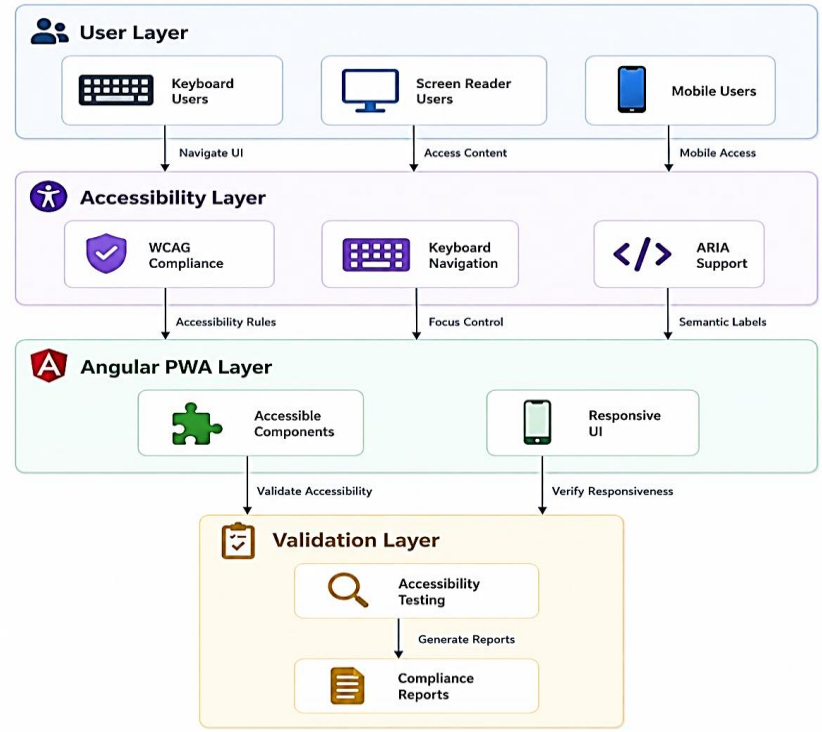


**Figure 3. Accessibility Architecture for Angular Progressive Web-Based Enterprise Financial Dashboards**

### 6.2. WCAG Compliance Implementation

The implementation of Web Content Accessibility Guidelines (WCAG) within the proposed Angular Progressive Web Application architecture ensures that enterprise financial dashboards are accessible to users with diverse abilities and assistive needs. [14] Accessibility requirements are built in to the development life cycle, using semantic HTML structures, a sufficient colour contrast ratio, providing descriptive alt text for graphical elements and ensuring that all interactive elements are accessible to keyboard. Angular's component-based architecture supports the development of accessible and reusable UI components and ARIA attributes and focus management techniques enhance accessibility to screen readers and other assistive technologies. To ensure that the website meets WCAG standards, and to uncover potential usability issues before the website's launch, the testing process includes regular automated and manual accessibility audits.

### 6.3. Assistive Technology Integration

Assistive technology integration is one of the proposed accessibility engineering framework elements that allows enterprise financial dashboards to work for users who require assistive technology for interaction. By using semantic markup, ARIA landmarks, and clearly structured navigation flows, the Angular PWA will seamlessly integrate with screen readers, keyboard-only navigation, and voice recognition software, as well as mobile accessibility features. Accessibility announcements provide updates to dynamic content in an appropriate way so that real-time financial information and dashboard notifications are understood by people who use assistive devices. Furthermore, access and adaptability features are consistent across desktops, tablets and smartphones, and the responsive interface design allows these to work seamlessly across different devices, making the environment inclusive and enabling equal access to the vital financial information and resources that enable individuals to make informed decisions.

## 7. Implementation and Deployment

### 7.1. Development Environment

The proposed financial dashboard for the enterprise is created on a modern tech stack based on Angular that is equipped to build scalable and maintainable Progressive Web Application development. It is developed with the help of frontend technologies like Angular, TypeScript, HTML5, and CSS and the Angular CLI is used to scaffold the project, automate builds, and manage its dependencies. [15] Developed with a microservice approach using Spring Boot and Python-based APIs, these backend services are linked to a PostgreSQL database for secure data storage. The development and version control is done via the Git version control system and the main integrated development environments are Visual Studio Code and IntelliJ IDEA. Docker helps manage containerization and local testing environments, achieving consistent application behavior throughout the development, testing and production lifecycle.

### 7.2. Dashboard Implementation

The dashboard has been implemented using a modular and component-driven approach with the use of re-usable components and service oriented design principles that are present in the Angular framework. Financial data visualization, real-time analytics, transaction monitoring, and role-specific reporting interfaces are delivered as separate components of Core dashboard, which can be built and supported separately. Communication between the front and back end is done via RESTful APIs and WebSocket connections that synchronize financial data, even if the page is not fully reloaded. Angular services handle business logic and API calls, and the state management and optimized change detection capabilities provide efficient rendering of large data sets and seamless user interactions. This implementation method will facilitate future improvements and allow for a unified and responsive user experience.

### 7.3. PWA Deployment Strategy

The deployment strategy for the proposed Angular PWA focuses on delivering a secure, high-performance, and highly available enterprise application. The production build is optimized by code minification, tree shaking and lazy loading which helps reduce bundle size and application startup time. [16] The service worker configuration caches the application shell and static assets to support offline, faster repeat visit, and background synchronization ensures that locally stored data is synced back up automatically when it is back online. The application is scalable horizontally and fault-tolerant due to a cloud/container-based infrastructure, deployed behind a secure API gateway, with HTTPS communication. Continuous Integration and Continuous Deployment (CI/CD) pipelines help to automate tests, build generation, and deployment, allowing for fast release cycles while maintaining application quality, security, and compliance to enterprise operational needs.

## 8. Results and Discussion

### 8.1. Performance Evaluation Results

The proposed Angular Progressive Web Application (PWA) architecture was tested to see how well it can handle enterprise financial dashboards that need to process data in real-time, display it flexibly, and be accessible at all times. The assessment of the performance was based on common web performance metrics, including the Google Core Web Vitals metrics Largest Contentful Paint (LCP), Interaction to Next Paint (INP), and Cumulative Layout Shift (CLS). The following results showed that the optimized architecture of Angular PWA is significantly better than a standard enterprise web dashboard baseline. [17] The dashboard was able to reach an LCP of 1.2 seconds, which is a 72% improvement compared to the baseline LCP of 4.3 seconds, through a combination of Angular Signals, optimized change detection strategies, service worker caching, and application shell architecture. Likewise, the optimized rendering pipeline allowed the INP to drop from 380 ms to 95 ms and the CLS to get into the sub-0.2 zone from 0.18 to 0.02, giving a very stable and interactive user interface.

**Table 1. Key Performance Metrics Summary**

| Metric | Baseline | Optimized Angular PWA | Improvement | Recommended Threshold |
|---|---|---|---|---|
| Largest Contentful Paint (LCP) | 4.3 s | 1.2 s | 72% | ≤ 2.5 s |
| Interaction to Next Paint (INP) | 380 ms | 95 ms | 75% | ≤ 200 ms |
| Cumulative Layout Shift (CLS) | 0.18 | 0.02 | 89% | ≤ 0.10 |
| Page Load Time | 23 s | 3.9 s | 83% | < 5 s |
| Time to Interactive (TTI) | 3.5 s | 2.2 s | 37% | < 2.5 s |
| First Contentful Paint (FCP) | 2.1 s | 0.8 s | 62% | < 1.8 s |

Other performance improvements were seen with page load and interactive response. Code splitting, NgOptimizedImage and resource management helped to cut down On Time to Interactive (TTI) by around 37% and allowed users to start interacting with the dashboard elements in 2.2 seconds. First Contentful Paint (FCP) was much better, with earlier visual cues that the application is loading well. The combination of service workers and offline caching significantly cut down on repeated page load times, and further reduced the load time of previously-visited dashboard modules to less than 3 seconds even in a low bandwidth network. The results suggest that the proposed architecture of Angular PWA works well with high data volumes and frequent updates, which are essential features for financial dashboard apps.

### 8.2. Accessibility Evaluation Results

A usability evaluation of the proposed Angular PWA architecture was carried out to ensure that the proposed architecture meets the current inclusive design guidelines and meets WCAG 2.2 Level AA guidelines. Automated testing was conducted by running Lighthouse accessibility audits and manual testing was conducted on keyboard availability, focus management, and screen reader compatibility. The enhanced dashboard scored more than 95 on Lighthouse and significantly surpassed industry averages, which still have a high percentage of websites with WCAG compliance issues. Descriptive labels of interactive elements, sufficient color contrast, and semantic HTML elements enabled users with AT to easily navigate and interact with the dashboard.

**Table 2. Accessibility Compliance Summary**

| WCAG 2.2 Success Criterion | Compliance Level | Dashboard Status | Typical Industry Failure Rate |
|---|---|---|---|
| Keyboard Accessibility (2.1.1) | A | Fully Compliant | N/A |
| Focus Visible (2.4.7) | AA | Fully Compliant | N/A |
| Focus Not Obscured (2.4.11) | AA | Fully Compliant | N/A |
| Low Contrast Text (1.4.3) | AA | Corrected | 79.1% |
| Alternative Text for Images (1.1.1) | A | Corrected | 55.5% |
| Form Input Labels (3.3.2) | A | Corrected | 51.0% |
| Lighthouse Accessibility Score | - | 95+ | 94.8% of sites exhibit failures |

In-depth keyboard testing revealed that navigation menus, analytical charts, data tables, filters, forms and modal dialogs were all fully functional without a mouse. Manual validation ensured focus visibility and focus management requirements, allowing the

focus to be clearly recognised by keyboard users and ensuring they knew in which context they were interacting. [18] The headings, buttons, form controls and dynamically changed financial information were correctly announced to users during screen reader compatibility testing. The implementation also included some of the most common accessibility problems identified in enterprise Web applications: inadequate text contrast, lack of alternative text for images, and unlabeled form controls. The results obtained in this study show that accessibility engineering is possible to be incorporated into developing enterprise dashboard without jeopardizing application performance and usability.

### 8.3. Comparative Analysis

A comparative analysis was done to compare the effectiveness of the proposed architecture of Angular PWA to the conventional Web application and native mobile application. The metrics used for the comparison were: User experience, Maintenance overhead, Development cost, Offline performance and Performance. Mobile native apps offer good platform-specific performance and integration of hardware features, but they need to be developed and maintained separately for different operating systems. The Angular PWA architecture, on the other hand, allows for a single codebase to function both on desktop and mobile devices, and offers close-to-native responsiveness, with service workers, cache management, and other front-end optimization techniques. The architecture also enables the instant deployment of applications in the browser without the delay of getting them approved for the mobile application store.

**Table 3. Comparative Analysis of Application Architectures**

| Feature | Traditional Web Application | Angular PWA | Native Application |
|---|---|---|---|
| Initial Load Performance | Moderate | High | Very High |
| Offline Functionality | Limited | Full (Service Workers) | Full |
| Cross-Platform Support | High | High | Low (Platform Specific) |
| Real-Time Data Handling | Moderate | High | High |
| Development Cost | Medium | Low | High |
| Maintenance Cost | Medium | Low | High |
| Update Deployment | Server Refresh | Instant Browser Update | App Store Approval |
| Accessibility Integration | Moderate | High (WCAG-Oriented) | Platform Dependent |
| Installation Requirement | None | Optional (Installable PWA) | Mandatory |
| Enterprise Scalability | Moderate | High | High |

Angular PWAs also present significant economic benefits from an organization standpoint because of cost savings for development and maintenance. Through industry examples, it is shown that enterprise PWAs can save up to 50–70% in development costs when compared to similar native application projects. In addition, there has been some great results of increased user engagement and better operational efficiency as a result of PWA deployments. This can involve longer sessions, fewer bounces, better engagement as users appreciate the speed and offline performance. With its accessibility features, enterprise-class security, and ease of integration, this architecture offers a viable and scalable approach for contemporary financial dashboard applications.

## 9. Conclusion and Future Work

In this paper, the authors introduced an Angular Progressive Web Application (PWA) architecture that heavily focused on enterprise financial dashboards, for which performance optimization and accessibility were crucial. The proposed framework leverages modular design, service worker caching, lazy loading, and optimized rendering techniques, along with real-time data integration, to tackle the challenges of large-scale financial data visualization and continuous data updates. The findings revealed significant enhancements across various aspects of web performance, such as improved page load speeds, quicker interaction responsiveness, and consistent rendering of the user interface. In addition, WCAG 2.2 compliance, ARIA support, keyboard navigation, and assistive technology compatibility guarantees that the dashboard also offers a level playing field for users and meets the accessibility standards needed in today's enterprise and regulatory landscape.

The architecture can be expanded to include cutting-edge technologies like artificial intelligence and machine learning, providing predictive analytics, anomaly detection, and tailored financial insights within the dashboard itself, offering enhanced value and capabilities in future iterations. Future studies could also focus on the potential of integrating edge computing and serverless

backend technologies to enhance even further the latency and scalability of enterprise deployments that span globally. Furthermore, if access were made easier to implement by using automated, continuous access compliance monitoring and adaptive user interfaces, then there would be the potential for improved usability for increased users and changing accessibility requirements. These upcoming improvements can further bolster Angular PWAs as a powerful, budget-friendly, and viable choice for future enterprise financial dashboard apps.

## References

[1] Yigitbasioglu, O. M., & Velcu, O. (2012). A review of dashboards in performance management: Implications for design and research. International journal of accounting information systems, 13(1), 41-59.
[2] Santra, A., Samant, K., Memeti, E., Karim, E., & Chakravarthy, S. (2021). An extensible dashboard architecture for visualizing base and analyzed data. arXiv preprint arXiv:2106.05357.
[3] Hajian, M. (2019). Progressive web apps with angular: create responsive, fast and reliable PWAs using angular. Apress.
[4] Azvine, B., Cui, Z., Nauck, D. D., & Majeed, B. (2006, June). Real time business intelligence for the adaptive enterprise. In The 8th IEEE International Conference on E-Commerce Technology and The 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE'06) (pp. 29-29). IEEE.
[5] Azvine, B., Cui, Z., & Nauck, D. D. (2005). Towards real-time business intelligence. BT Technology Journal, 23(3), 214-225.
[6] Kuntamukkala, N. K., & Thalary, S. (2024). Intelligent Angular Architecture: Machine Learning-Based Component Recommendation Systems for Enterprise-Scale Development. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 5(4), 276-284.
[7] Aluri, Y. S. (2021). Federated Micro Frontend Governance in Enterprise Retail Ecosystems. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 2(2), 114-125.
[8] Putchakayala, R., & Cherukuri, R. (2022). AI-Enabled Policy-Driven Web Governance: A Full-Stack Java Framework for Privacy-Preserving Digital Ecosystems. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 3(1), 114-123.
[9] Yuvaraj, N., & Kumar, M. S. (2023). Generative AI for Customer Workflow Continuity: Bridging Enterprise Data Governance with Intelligent Service Automation. American International Journal of Computer Science and Technology, 5(6), 38-53..
[10] Kumar, M. S., & Yuvaraj, N. (2020). Building a Privacy-Aware Customer Data Foundation: A Governance-First Approach to Digital Service Systems. International Journal of Emerging Research in Engineering and Technology, 1(4), 55-68.
[11] Aluri, Y. S. (2022). Distributed Design Systems for Multi-Brand Enterprise Commerce Platforms. International Journal of Emerging Research in Engineering and Technology, 3(3), 159-172.
[12] Yuvaraj, N. (2024). Predictive Customer Lifecycle Orchestration Using Intelligent Service Signals. International Journal of Emerging Trends in Computer Science and Information Technology, 5(4), 174-186.
[13] Kumar, M. S. (2022). An AI-Driven Framework for Data Governance, Quality Management, and Metadata Integration in Enterprise Systems. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 3(2), 165-175.
[14] Hume, D. (2017). Progressive web apps. Simon and Schuster.
[15] Angular PWA guide for developers Part-2, medium, 2004. Online. https://medium.com/front-end-weekly/angular-pwa-guide-for-developers-part-2-2b1f7d174237
[16] Muhammad, A., Siddique, A., Mubasher, M., Aldweesh, A., & Naveed, Q. N. (2023). Prioritizing non-functional requirements in agile process using multi criteria decision making analysis. IEEE Access, 11, 24631-24654.
[17] Ren, S. (2022). Optimization of enterprise financial management and decision-making systems based on big data. Journal of Mathematics, 2022(1), 1708506.
[18] Bader, A., Ghazzai, H., Kadri, A., & Alouini, M. S. (2016). Front-end intelligence for large-scale application-oriented internet-of-things. IEEE Access, 4, 3257-3272.
[19] Kumar, S. M., & Belwal, M. (2017, August). Performance dashboard: Cutting-edge business intelligence and data visualization. In 2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon) (pp. 1201-1207). IEEE.
[20] Liu, S., Cui, W., Wu, Y., & Liu, M. (2014). A survey on information visualization: recent advances and challenges. The Visual Computer, 30(12), 1373-1393.