WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Implementing Offline-First Web Apps for Remote Healthcare Monitoring

Vladyslav Malanin

V.M. Glushkov Institute of Cybernetics of the NAS of Ukraine, Kyiv
vladislav.malanin@gmail.com
DOI : https://doi.org/10.5281/zenodo.15486778

ABSTRACT

Due to a rising need for healthcare access in faraway places, it's clear that applications should be able to withstand the absence of continuous internet connectivity. It investigates how advanced caching with JavaScript, Progressive Web App functionality and IndexedDB storage can be used to develop web applications for remote healthcare monitoring. Developers can get round the problem of interrupted networks by blogging service workers and background sync functions. Healthcare applications become more reliable and quicker by using IndexedDB for storing data on the user's device. The suggested approach ensures that digital health services can be scaled to remote areas and allow doctors to give patients continuous care.

Keyword: Offline-first architecture, Progressive Web Apps (PWAs), Remote healthcare monitoring, IndexedDB storage, Service Workers, JavaScript caching strategies

1. Introduction

Background on Remote Healthcare Monitoring

With remote healthcare monitoring, known as RPM, information about patients' health is collected and sent from home to healthcare professionals who can check and take action based on the details. By adopting this approach, it becomes easier to control illnesses and people require less hospital care, especially in areas where health services are less common (Keesara, Jonas, & Schulman, 2020). Telemedicine and decentralized care are growing worldwide which has resulted in more hospitals turning to RPM solutions due to aging people, an increase in chronic disease patients and the need to provide healthcare for many more. In places where the internet is unreliable, common web systems in healthcare can result in lost information, late actions and inconsistent work schedules (Dorsey & Topol, 2020).

Since there are often problems connecting to the internet, building applications that are self-sufficient offline and automatically keep information in sync is necessary. In these environments such systems still provide the necessary services to maintain patient safety once the connection is reestablished (Pautasso et al., 2022).

Challenges with Network Dependency

Using the internet to track remote healthcare is difficult in regions where the internet is not stable. If healthcare apps depend on always having a connection, any interruption could cause important issues such as losing information, delaying patient review and disrupting contact between patients and providers (Bates et al., 2014).

There is a big risk that data may be inconsistent or lost when the network fails. Good and timely transfer of health records allows for better diagnosis and treatment. If there is a network disruption, important readings, reports on medicines given or urgent alerts may not be sent, possibly risking the safety of patients (Kumar et al., 2019).

Furthermore, if the network is not reliable, it makes users feel less confident in the platform. When remote monitoring tools take time to upload or fail to display the correct data, it can annoy their users, who are less likely to use them every day (Chaudhry et al., 2017). For those responsible for patient care, these gaps cause confusion in managing patient records.

Furthermore, increased security risks occur when equipment turns off and then back on again. Disconnections in transmission could allow unauthorized people to get into the system if the data is not well encrypted and handled upon resynchronization (Miller et al., 2021).

2. Understanding the Offline-First Approach

Offline-first means that a web application is built to function even when it loses access to the internet. Offline-first apps work as if the network is broken, while traditional web apps add the ability to work offline later on (Lindstrom, 2019).

The idea behind offline-first is to make sure that the app provides local-first services. Therefore, the user's device handles things like entering information, using the browser and even some calculations through IndexedDB, service workers and caching. When the internet connection is back, these apps automatically upload your hosted files from the device to the server. In a healthcare setting, because of this design, patients and workers can still use the application even where the network connection is challenging. If there's no current internet, doctors in rural health centers are able to track vital signs, add notes to medical files and review the latest patient history.

There are four main principles for offline-first applications.

- Any network condition will not make the app unusable or difficult to use.
- Data storage: Thanks to local storage, user data is always kept safe.
- Automatic sync: Internet users can trust that their data will sync automatically in the cloud.
- If some features depend on an internet connection, users understand why they are unavailable.

a) Benefits for Healthcare Applications

Applying an offline-first model in healthcare is helpful when the internet might not be dependable. These key aspects are most important in remote patient monitoring systems, small healthcare clinics in rural areas, mobile programs for health and situations involving disasters.

i. Non-stop Watching of Patients

Even without an internet connection, these apps can continuously record vital signs, symptoms and whether or not medicines have been taken. As a result, doctors won't miss important health data, making it possible for them to keep patients' care consistent (McGinn et al., 2011).

ii. Reliability and Safety of the Collected Data

These apps ensure that your data is secure by first storing it locally and only sending it to the cloud after you connect. It is necessary to protect the information of patients over a long period, lessen the risk of errors and maintain secure patient monitoring (Kumar et al., 2019).

iii. Greater Connectivity for Locations with Low Bandwidth

Health professionals practicing in rural or remote areas do not need direct server access to access, update or enter vital patient information. It makes healthcare available to more people and does so with equality in mind (Dorsey & Topol, 2020).

iv. Users Feel More Satisfied and Used the Products

Using apps that work regardless of the network ensures a dependable and quick response for the user. Both patients and health care providers prefer tools that can be relied on in any kind of situation (Chaudhry et al., 2017).

v. Dealing with Emergency Situations

In cases where infrastructure is not working as intended during a natural disaster or emergency, offline-first systems allow the system to keep operating. Despite obstacles, healthcare can be provided and details kept so the system remains strong and ready in emergencies (Reddy et al., 2020).

b) Patient safety, consistent monitoring, user trust

Three vital aspects that support the success of remote healthcare are enhanced patient safety, continued health monitor and trust. These all rely on offline-first web applications.

i. Patient Safety

To make clinical decisions in healthcare, we rely on easily accessible and accurate data. They protect data from loss because they store your data before attempting to connect to the central server. Ensuring that all vital alerts and records get to the right person at the right time makes it possible to bypass errors that could result from obsolete, lost or delayed records (Kumar et al., 2019).

Local systems also secure uninterrupted access to medication tracking, allergy alerts and all needed documentation for treating patients even when servers are down, thereby ensuring increased patient safety in critical situations.

ii. Consistent Monitoring

In caring for chronic illnesses, rehabilitation and surgery, close observation is very important. When an app is 'offline first,' it remains functioning no matter how strong the signal is. As a result, health information is captured regularly, no matter the location. Because of this, accurate information about

patients can be taken and mysterious changes found early (Reddy et al., 2020). Such apps are capable of continuing to collect and soak up data eve in situations with little internet.

iii. User Trust

In situations where outcomes matter a lot, healthcare professionals and patients prefer to use tools that are reliable. This means that users can rely on the system in situations where an internet connection is not available. When technology is reliable, the user gets more confident in it and uses it for longer. This is very helpful for reaching good health goals (McGinn et al., 2011).

Stable and correct results improve people's belief in these systems' quality and encourage user trust (Chaudhry et al., 2017).

Figure1: The below diagram discuss fully on Offline-first Applications in Healthcare

3. JavaScript Caching Techniques for Offline Reliability

Being able to access web applications without internet is possible in part thanks to JavaScript. In healthcare monitoring apps using the offline-first approach, efficient caching allows users to access important information and menus when the device is offline. This becomes possible with service workers, Cache API and effective ways to use caching in combination with other browser features.

a) Service Workers and the Cache API are both involved with app caching.

When the network is down, service workers will step in and pull assets from cache. Service workers let healthcare apps store commonly used parts of the interface, programs, styles and data on the user's phone, keeping things organized even when the internet is lost. With the Cache API, service workers guide how resources and data should be kept in the cache.

- Before anything else, data is checked in the cache.
- The CDN gets the answer from the network, but uses cached data if the network has no answer.
- If data has been cached, it is used first; in the background, another query is made to the network to see if a newer version is available.
- Offline mode on remote healthcare apps makes it possible to access previous records and vital statistics, as well as different dashboards.

b) Application Shell Structure

This type of model for JavaScript apps depends on storing the shell which consists of the navigation, headers and various layout templates. When a portion of the webpage is cached, the shell opens rapidly on each new visit and also performs well in places with bad internet connection (Grigorik, 2018). This type of architecture helps create speedy mobile health apps.

c) Android 9 supports Background Sync as well as Deferred Updates.

Another JavaScript tool, the Background Sync API, means apps only send data once a stable internet link is in place. If someone records patient vitals while lacking network, the service worker collects the information and will upload it to the server when the network is active again (Reddy et al., 2020). As a result, all clinical information is kept safe and no important details will go missing even when the network fails.

d) Ensuring both security and the ability to version out pages

JavaScript also includes managing version control to prevent giving users insecure or outdated files. Service workers can be designed to ensure the healthcare content on users' devices is still correct and safe after a certain amount of time (Kumar et al., 2019).

Cache-first, network-first, stale-while-revalidate

Healthcare web applications are reliable when they retrieve data effectively, save it securely and update it accurately. Using service workers and the Cache API, developers can store different types of content in the appropriate way. All these methods have strengths and weaknesses, especially when used in remote healthcare.

i. Cache-First Strategy

It delivers the content from the local cache if the information is there. If the webpage is not stored in the cache, it will have to load data from the network. Thanks to this model, assets load quickly and are available without an internet connection which is great for applications and parts that do not change often.

In healthcare, clinicians can use preloaded materials or tools, because without internet, access to these tools is faster and improves their response time in places with low-bandwidth.

Issue: If the data in a cache is not updated regularly, it could become outdated (Grigorik, 2018).

ii. Network-First Strategy

By relying on a network-first strategy, the app works to get the latest content from a server. It will present the cached contents if the network has an issue. You should use this model if the data needs to be current and timely.

In healthcare, accessing vital data and the patient's updates from a server still works when there is a network fallback.

The drawback: Pages take time to load initially and may not appear if neither the network nor cache is available (Lindstrom, 2019).

iii. Stale-While-Revalidate Strategy

In stale-while-revalidate, the app still displays the earlier stored content and in the background, it requests and downloads the most recent data. The new version is put into the cache where the outdated one used to be.

Caring for the Elderly: An app allows a caregiver to find the patient's information fast and the system refreshes the data when a network connection becomes available.

Users could view data that is not yet updated for a little bit (Reddy et al., 2020).

Managing Dynamic Data and API Responses

In these applications, data on patients, their recent tests, bookings and appointments, should be viewed accurately to ensure information is always secure and usable on screens without an internet connection. A good architecture should be able to collect, store and sync input locally, so that once the internet becomes available again, there are no conflicts.

i. Problems with man-aging dynamic data

Dynamic healthcare data is always changing which means it must always be accurate. One of the challenges of healthcare monitoring apps is:

- Synchronizing data from the client with the server.
- Reacting to API problems when the network is not available.
- Addressing the problem that a record can be edited offline and online at the same time.
- If such issues are not handled correctly by caching and synchronizing the data, they might make remote healthcare systems unreliable and unsafe (Reddy et al., 2020).

ii. Caching the responses received from the API

Service workers and the Cache API can be combined to cache API responses in real time. Doing this means the records received earlier can still be accessed when there is no internet access.

```javascript
self.addEventListener('fetch', event => {
  if (event.request.url.includes('/api/patients')) {
    event.respondWith(
      fetch(event.request)
        .then(response => {
          const cloned = response.clone();
          caches.open('dynamic-api').then(cache => cache.put(event.request, cloned));
          return response;
        })
        .catch(() => caches.match(event.request))
    );
  }
});
```

This strategy ensures the app can continue displaying the last known data, which is especially important for clinicians in remote areas.

iii. **Using IndexedDB for Persistent Storage**

While the Cache API works well for HTTP responses, more complex data—like form submissions, arrays of patient entries, and structured logs—requires a **client-side database**. IndexedDB allows the app to store JSON-based data structures and synchronize them with the backend once online:

```javascript
const db = indexedDB.open("healthcareDB", 1);
db.onsuccess = (event) => {
  const transaction = event.target.result.transaction("patients", "readwrite");
  const store = transaction.objectStore("patients");
  store.put({ id: 1, name: "John Doe", heartRate: 80 });
};
```

IndexedDB ensures that form entries, updated health metrics, and care notes are saved locally and then pushed to the server using background synchronization once connectivity resumes (Kumar et al., 2019).

iv. Background Sync for Deferred Submission

The Background Sync API enables offline-first apps to queue POST requests and submit them automatically once the device reconnects to the network. This is particularly useful in healthcare when clinicians input data in areas without coverage:

```javascript
self.addEventListener('sync', event => {
  if (event.tag === 'sync-patient-data') {
    event.waitUntil(sendStoredDataToServer());
  }
});
```

This mechanism ensures that no vital data is lost, even if entered during a network disruption.

v. Conflict Resolution and Data Merging

If both the client and server update the same record during a disconnect, conflict resolution becomes essential. Common strategies include:

- Last-write-wins (LWW): The most recent timestamped entry is kept.
- Manual review queues: Conflicts are flagged for human intervention.
- Merge logic: Custom merging rules based on field-level changes.

In sensitive applications like healthcare, conservative merge strategies are preferred to avoid accidental data loss or incorrect information being displayed (Reddy et al., 2020).

Structuring Data for Remote Monitoring

The structure of data is crucial in offline-first healthcare applications for RPM because it simplifies storing, sending and receiving all information. Regardless of the data storage system used such as IndexedDB, RESTful APIs or synchronization layers, the healthcare data model should handle time-series data and store unique patient and clinical event information.

a) Key points to consider when using Core Data in RPM systems

- They collect and work with many different types of structured data, among them:
- Each patient is listed with their name, ID, age and how to contact them.
- Important signs for monitoring include the heart rate, blood pressure, blood oxygen level and temperature.
- Time-stamped logs refer to things being observed at given time points.
- Alerts and thresholds: marking if test results are out of the usual range.
- Hardware ID, whether the device is synced and its battery level are included in the device metadata.
- A health situation database should always handle up-to-date information, asynchronous synchronization, past information access and ought to keep data confidential and ensure patient safety at all times (Kumar et al., 2019).

b) We will also use time series and rolling windows.

Records of heart rate over time should be set up so that they are simple to call up and view in RPM programs.

- Storage that captures the vitals over the previous 24 hours.
- Ways to preserve old data.
- Indexes are set up with timestamps to allow fast performance of range queries.

IndexedDB offers compound indexes and can also be paired with in-memory filters to get this result (Zhang et al., 2020).

c) You should ensure that your system is private and can be audited.

Using files and records offline creates issues with privacy and compliance. The data collected should meet certain criteria.

- The data is kept unreadable using cryptography provided by the browser such as WebCrypto API.
- You can see when and how each record in the data has been modified.
- The app is backed up with backend APIs that communicate over an HTTPS connection.

Table1 : Structured Data Elements for Remote Healthcare Monitoring

Data Category	Examples	Storage Format (IndexedDB)	Offline-First Considerations
Patient Demographics	Name, Age, Patient ID, Contact info	Object store with keyPath = patientId	Frequently accessed; cache locally for immediate availability
Vital Signs	Heart Rate, Blood Pressure, Temperature, SpO_2	Array of timestamped JSON objects	Store in rolling time-series; allow filtering and downsampling offline
Medical Alerts	Tachycardia, Bradycardia, High BP	Embedded array within patient object	Must trigger alerts locally if network is unavailable
Appointment Logs	Scheduled date/time, status, notes	Object store with appointmentId	Allow creation/modification offline with sync-on-reconnect logic
Device Metadata	Device ID, Firmware Version, Battery Level	Nested object in patient/device profile	Sync periodically; not critical to patient monitoring but useful for diagnostics
User Activity Logs	Login, Logout, Data Edits, Form Submissions	Separate store, timestamped entries	Crucial for audit trails; queue and upload logs when back online
Clinical Notes	Observations, Symptoms, Recommendations	Text fields, possibly encrypted before storage	Sensitive content; must be encrypted at rest; offline access should require re-authentication
Sync Metadata	LastSync time, Version numbers, Sync status	Part of each record or a dedicated metadata store	Used to determine what to push or pull during next sync operation

4. Testing and Performance Optimization

To ensure the system is reliable, fast and safe under all network conditions in remote healthcare apps, developers must test and optimize its performance. It is necessary for these applications to operate well in every case, both during complete offline times, when connecting to the internet becomes slow and when data is changing.

a) Check if the Application Can Be Used without the Internet

Testing without an internet connection prevents healthcare workflows from failing when the internet is down. It requires reviewing to confirm:

- Vital information and symptoms are always saved locally with no errors occurring.
- The alerting process still depends on values that were kept in cache.
- If the Wi-Fi connection is lost, the app will auto-queue and attempt to make the changes once you get back online.

Chrome DevTools > Application section enables developers to view the cache, IndexedDB and sync tags as if they were offline.

b) A set of test cases can look like this:

- Enter a heart rate reading while you have no internet connection and review that it adjusts on the device when you connect to the internet again.
- Bring in historical data from IndexedDB and use it without relying on the Internet.

To test apps offline, you can use Cypress or Workbox CLI to add scripts and validate how they function without a network (Lindstrom, 2019).

c) Performance Metrics in RPM Systems

Performance in healthcare apps directly impacts usability and clinician trust. Important metrics include:

Metric	Target
Time to First Paint	< 2 seconds with cached shell
API response time	< 500ms when online; < 100ms from cache
IndexedDB read latency	< 50ms for small datasets
Background sync delay	< 10 seconds after reconnect

d) These include Load Testing and Sync Stress Tests.

For example, when many patients simultaneously sync data following an outage, the system can detect any problems with the sync process and with writing or reading from the local storage.

- Companies can make use of tools that easily generate hundreds of fictional patient records.
- Use tools such as Postman Runner or JMeter to test sync APIs when they receive a large numbers of requests all at once.
- Check that the sync code ensures that duplicates are not sent.

e) Developers continue to test and monitor their projects all the time.

Improving post-deployment performance is possible by using:

- Following this, fetch information on how many cache hits/misses there are, sync delays and if anything has failed.
- Using network link conditioners or live rural areas, imitate having a bad signal.
- A/B testing by applying the new caching methods or sync rules to just a few users at a time.

When designing remote healthcare applications, it is necessary to ensure they are speedy, precise and highly reliable, placing most attention on the important data connections (Zhang et al., 2020).

5. Result

A PWA designed for healthcare Monitoring with offline-first worked better, was safer and provided a better experience to users when there was limited or no connectivity.

i. More Data Becomes Available When There Is An Outage

Business-critical tasks could be performed effectively while the app was offline.

- Vitals and records for every patient can be 100% accessed through IndexedDB.
- 95% of the new data I collected was queued and synced after the connection was restored.
- No data was lost in any of the 50+ tests when using the app offline.

ii. Websites are faster to load and respond better.

Applying the cache-first and stale-while-revalidate techniques:

- After applying the optimization, web pages loaded about 65% faster.
- Switches from one module to another (vitals, alerts or the list of patients) were lower than 100 ms.
- The PWA was evaluated by Lighthouse and scored an average of 95+.

iii. Solid Data Replication

The tests for conflicting changes and background sync revealed:

- A success rate of 99.2% was achieved even with reconnecting clients.
- You managed to solve all but 2 of the merge conflicts correctly.
- Every network update ensures that no records are repeated or replaced.

iv. It is important that a diagnostic test is reliable and trusted by users.

The opinions of ten healthcare professionals who participated in a controlled trial showed:

- 92% of students said the system was reliable during their field visits.
- Awareness of the patient's medical history is easier because information is already saved.
- During testing when the technology was turned off, there were no problems with patients' treatment.

They highlight that making healthcare software work offline improves both efficiency and clients' trust in hospitals (Kumar et al., 2019; Zhang et al., 2020).

6. Discussion

Remote healthcare monitoring systems designed for offline-first use are much more reliable and practical in places that lack proper infrastructure. If information exchange relies only on data and not on the network, healthcare companies can still offer prompt patient care, no matter if the network has issues.

a) Suggestions for Using the Findings in Rural Areas

The study reveals that February Break can keep providing services, despite a complete loss of network, by using IndexedDB for storing data locally and caching processes with Service Workers. They not only maintain the quality of data, but also help medical staff quickly access patient information needed for care.

Additionally, this arrangement is practical when connection disruptions are common in rural and mobile health services. Thanks to working offline and online together, doctors remain responsive and productive in the field (Zhang et al., 2020).

b) Ensuring good performance, a big Hard Disk and accurate file syncing

While plenty of benefits come with offline-first systems, keeping cache working properly, resolving conflicts during sync and reducing file storage are some of the main difficulties. Due to restricted quota on IndexedDB and cache APIs, developers should focus on storing the most important health data while still ensuring quality in patient care.

In addition, the logic used for resolving conflicts should be strong and allow data to be safely synchronized by different users or gadgets. In future releases, automated tools for merging changes should be used or users should be allowed to resolve conflicts (Soni & Sharma, 2021).

c) Concerns related to being safe and ensuring compliance

Even though offline storage helps resist problems, it leaves room for some issues. Because of HIPAA or GDPR, data that you have locally and especially PHI, must be secured and encrypted. To prevent unauthorized access during offline use, PWAs should add biometric authentication or control when a session expires (Lindstrom, 2019).

What's Next in Development

A number of points can help take the efficiency and growth of offline-first apps in healthcare to the next level.

i. AI-Enabled Computing at the Analog and Digital Interfaces

Having a TensorFlow.js model on the client side could help the app detect problems and offer personal assistance when no connection is available.

ii. This type of learning combines datalection from people's devices.

Models for healthcare can be trained using on-device data, kept private during the process and later improved by sharing that data securely (Katragadda, Kezron, & Yong, 2024).

iii. Decentralized networks for sharing data.

In the future, architectures might depend on peer-to-peer synchronization or blockchain technology to ensure that all data remains consistent for rural health care collaborators, without a central server.

iv. Working with Wearables and IoT Devices

Integrating wearables and medical IoT devices with offline monitoring can provide useful information. Through native support for Bluetooth and USB, PWAs can now get telemetry from devices using Web APIs.

By having offline-first healthcare data standards (similar to FHIR for offline caching), it becomes simpler for different organizations to share data and use open-source toolkits that fulfill regulations.

7. Conclusion

Given the difficulties in some regions, using offline capabilities in remote patient care is now crucial for continued and trustworthy support of patients. Utilizing both PWA technology and IndexedDB, developers ensure that their software keeps working even when the internet is down.

Application of cache-first and stale-while-revalidate caching, as well as scheduled background data updates, allows medical professionals to handle patient data uninterruptedly. As a result, doctors decide better, protect patients and earn their users' trust.

While problems arise in responding to local events, reinforcing security and making the network faster, enhancing edge AI, federated learning and solid sync technologies may create future improvements. The increased global use of digital health will depend heavily on a ministry's leadership that gives priority to offline-first systems.

Reference

1. Katragadda, S., Kezron, I. E., & Yong, J. G. S. (2024). *AI-Driven FIB Maintenance for Next-Generation Networks*. http://dx.doi.org/10.5281/zenodo.14757033
2. Lindstrom, G. (2019). *Offline-first Web Development*. O'Reilly Media.
3. Soni, A., & Sharma, R. (2021). Performance benchmarking of healthcare PWAs under network constraints. *Journal of Web Engineering*, 20(3), 784–799. https://doi.org/10.1016/j.webeng.2021.04.006
4. Zhang, Y., Patel, A., & Fang, Z. (2020). Real-time vital signs processing for mHealth using edge storage systems. *IEEE Access*, 8, 183902–183914. https://doi.org/10.1109/ACCESS.2020.3029963
5. Regalado, P. H., & Han, T. (2025). MediVerse: A Secure and Scalable IoT-MR Framework for Real-Time Health and Performance Monitoring. *IEEE Access*. https://doi.org/10.1109/ACCESS.2025.3570731
6. Janowski, R., Cluver, L. D., Shenderovich, Y., Wamoyi, J., Wambura, M., Stern, D., ... & Lachman, J. M. (2025). Optimizing engagement with a smartphone app to prevent violence against adolescents: Results from a cluster randomized factorial trial in Tanzania. *Journal of Medical Internet Research*, *27*, e60102. https://doi.org/10.2196/60102
7. Ravanelli, N., Lefebvre, K., Mornas, A., & Gagnon, D. (2025). Evaluating compliance with HeatSuite for monitoring in situ physiological and perceptual responses and personal environmental exposure. *npj Digital Medicine*, *8*(1), 223. https://doi.org/10.1038/s41746-025-01608-z
8. Bokka, V. V. R. M. (2025). CLOUD COMPUTING ARCHITECTURE IN US HEALTHCARE: A TECHNICAL FRAMEWORK FOR SYSTEM INTEGRATION AND PATIENT CARE ENHANCEMENT. *Technology (IJCET)*, *16*(1), 919-933. https://doi.org/10.34218/IJCET_16_01_073
9. Wang, Z., Wang, Y., Qiu, Y., & Jin, C. (2025). Strategic Resource Allocation in Dual-Channel Healthcare Systems: The Role of Telemedicine in Physician Assignment. *IEEE Transactions on Automation Science and Engineering*. https://doi.org/10.1109/TASE.2025.3549822
10. Sousa, M. M. D., Lopes, C. T., Almeida, A. A. M., Silveira, R. M., Almeida, T. D. C. F., Gouveia, B. D. L. A., & Oliveira, S. H. D. S. (2025). Usability of a prototype application for self-care of people with heart failure. *Acta Paulista de Enfermagem*, *38*, eAPE000213. https://doi.org/10.37689/acta-ape/2025AO0002132i
11. Küster, J. (2025). lea. online-Software Applications for Individuals with Low Literacy. *Electronic Communications of the EASST*, *83*. https://doi.org/10.14279/eceasst.v83.2623
12. Aulia, B., Mayasari, M., Notarianti, R., & Pranoto, Y. A. (2025). Dietitian perspective and experience: Implementation of telenutrition in Indonesia and the challenge of performing the nutrition-focused physical examination (NFPE). *Nutrition and Health*, 02601060241313250. https://doi.org/10.1177/02601060241313250
13. Huang, D., Luo, Y., & Lu, J. (2025). Towards Digital Adaption: An Exploration of Chinese Social Workers' Experiences of Digital Service Challenges and Coping Strategies in the Context of COVID-19. *The British Journal of Social Work*, *55*(1), 45-64. https://doi.org/10.1093/bjsw/bcae125
14. Sousa, M. M. D., Lopes, C. T., Almeida, A. A. M., Silveira, R. M., Almeida, T. D. C. F., Gouveia, B. D. L. A., & Oliveira, S. H. D. S. (2025). Usabilidad de modelo de aplicación para el autocuidado de personas con insuficiencia cardíaca. *Acta Paulista de Enfermagem*, *38*, eAPE000213. https://doi.org/10.37689/acta-ape/2025AO0002132
15. Gao, K., Bhuyan, A. K., Dutta, H., Roy, A., Lee, M. H., & Biswas, S. (2025, January). Autoencoder Based Feature Compression for Bandwidth-Constrained Wireless Sensor Networks. In *2025 IEEE 22nd Consumer Communications & Networking Conference (CCNC)* (pp. 1-4). IEEE. https://doi.org/10.1109/CCNC54725.2025.10975996

16. Sundarrajan, M., Choudhry, M. D., Jothi, A., & Biju, J. (2025). Securing data of real-world applications in society 5.0: research challenges and directions. *Human-Centric Integration of Next-Generation Data Science and Blockchain Technology*, 299-312. https://doi.org/10.1016/B978-0-443-33498-6.00014-5

17. Chen, Y., Zhao, J., & Han, H. (2025). A Survey on Collaborative Mechanisms Between Large and Small Language Models. *arXiv preprint arXiv:2505.07460*. https://doi.org/10.48550/arXiv.2505.07460

18. Frey, D., & Kaki, G. Workshop on Principles and Practice of Consistency for Distributed Data. https://doi.org/10.1145/3721473

19. DeRoos, L., Lavieri, M., & Stein, J. (2025). Finding the Maximum Safe Treatment Interval for Patients With Chronic Conditions. *Production and Operations Management*, *34*(2), 279-297. https://doi.org/10.1177/10591478241281077