Journal of Artificial Intelligence and Engineering Applications

Website: https://ioinformatic.org/

15th February 2026. Vol. 5. No. 2; e-ISSN: 2808-4519

Web-Based Spare Parts Expenditure Recording Portal with Read-Only Pull from ERP Infor (PT CBI Case Study)

Alvito Kurnia Fahrio [1], Oman Komarudin [2], Intan Purmanasari [3]

[1,2,3]*Faculty of Computer Science, University Singaperbangsa Karawang*
alvitokurnia@gmail.com[1*]

Abstract

A companion portal for spare parts management was developed at the PT Century Batteries Indonesia (PT CBI) warehouse to address manual record keeping that is prone to discrepancies and difficult to trace. This article focuses on two key aspects: a checkout flow that validates expenditure amounts against Portal stock, and a reconciliation mechanism that pulls on-hand data from Infor's ERP read-only system to maintain ERP data integrity. The system was developed following the Extreme Programming (XP) methodology with short iterations and regular feedback from warehouse users. Black-box testing and UAT (User-to-User) testing demonstrated that the main flow functioned as expected; all assessed features were accepted with scores of 4.1–4.8. Consequently, discrepancies were detected more quickly and addressed through adjustments in Infor; the Portal then pulled back on-hand to ensure consistency. These results demonstrate that the checkout in Portal adjustment in Infor pulled back on-hand (read-only) pattern effectively reduces errors caused by manual recording while improving transaction traceability in the spare parts expenditure process in the warehouse environment.

Keywords: *warehouse management, spare part checkout, Infor ERP, read-only integration, Extreme Programming*

1. Introduction

Currently, many companies are turning to digital systems to increase work efficiency and reduce manual errors, including PT Century Batteries Indonesia (PT CBI) which is currently undergoing a digital transformation in its operational management.[1]However, observations show that the process of recording the expenditure of spare parts is still done manually, causing problems such as disorderly records, difficulty tracking history, and potential discrepancies between records and physical stock. Literature shows the problem of Inventory Record Inaccuracy (IRI) mismatch between physical stock and records that impacts warehouse/retail performance; a study of 37 stores (369,567 records) found 65% of stock records were inaccurate.[2]. Although previous research has discussed warehouse digitalization, many studies have designed and implemented new Warehouse Management Systems (WMS) built from scratch, rather than integrating existing systems.[3], [4]Therefore, there is still a research gap in system integration in companies that already have a primary read-only ERP system.

PT CBI itself uses two digital systems: Infor as the primary system for recording incoming stock, and Portal as an internal system developed by the IT team to support warehouse operations. Because the Infor system cannot be modified, recording outgoing goods is not available within it. Portal was then developed as a companion system that functions to record outgoing goods, display transaction history, and generate reports that help admins make adjustments in Infor. This system does not change stock data in Infor, but rather rereads (pulls) data after manual adjustments to maintain consistency between system and physical stock. Thus, Portal is designed to improve accountability and efficiency in recording outgoing goods without disrupting the company's main system.

Several previous studies have highlighted the importance of warehouse management systems, but not many have examined companion solutions for companies with read-only ERPs.[5], [6]. This gap makes this research important both practically and academically. The novelty of this research lies in the design of the Portal as an independent web system that focuses on recording goods expenditure with validation, history, and reporting integrated with the main system without changing it. The purpose of this research is to design, implement, and evaluate the performance of the Portal as a supporting system that improves efficiency, traceability, and accountability in the spare part expenditure process at PT Century Batteries Indonesia.

2. Theoritical Basic

2.1. Technologies Used in Warehouse Management Systems

CodeIgniter4 is a PHP framework with an MVC architecture that separates business logic and display, making web development more efficient and manageable. This framework supports various database types, including SQL Server, and is known for its lightness and speed. Research shows that using CodeIgniter in a web-based warehouse management system can improve operational efficiency and accuracy. SQL Server Management Studio (SSMS) is used to manage SQL Server databases with referential integrity and query optimization features that ensure data security and efficiency.Extreme Programming(XP) was chosen because it emphasizes intensive collaboration, short iterations, user involvement, and engineering practices such as continuous testing and refactoring to maintain software quality.[7]Through the stages of planning, design, coding, refactoring, testing, and release, XP has been proven to increase system reliability and software development efficiency.[8].

2.2. Data Synchronization in Warehouse Management System

Data synchronization in the warehouse management system ensures inventory accuracy and integration efficiency with external systems such as ERP and logistics.[9]This process is typically accomplished through API integration, middleware, real-time monitoring, and cloud technologies to maintain data consistency. Challenges arise in legacy systems that don't support direct modification, necessitating solutions such as batch processing or manual integration. In this study, a read-only pull approach was implemented, where the Portal only reads on-hand values from Infor for validation and reporting without modifying the data, allowing for one-way alignment.

2.3. Database

A database is a collection of data interconnected through relational keys and managed by a database management system (DBMS) to facilitate the process of storing, retrieving, and managing information. Databases are widely used in various industries, from fraud detection with machine learning algorithms, NoSQL-based document management, to entertainment and gaming systems serving millions of users simultaneously. In the context of modern data management, relational databases such as SQL Server excel at maintaining data consistency through constraints (PRIMARY/FOREIGN KEY, CHECK/UNIQUE) and trigger support to enforce cross-table business rules.[10]However, recent trends show the adoption of NoSQL to support the scalability and flexibility of logistics systems, as well as SQL–NoSQL combinations to balance data structure and processing efficiency.

2.4. Unified Modeling Language (UML)

Unified Modeling Language(UML) is a visual modeling language used to describe business processes, workflows, and relationships between system components. Use Case Diagrams show user interactions with the system. Activity Diagrams depict the flow of activities, and Sequence Diagrams describe the sequence of interactions between objects.[11]Class diagrams map the structure of classes and their relationships, while state diagrams show the changing state of objects throughout the system's lifecycle. UML helps developers understand system requirements holistically and ensures efficient and structured software design.

2.5. Software Testing

Software testing is an important stage to ensure the quality and suitability of the system; this is in line with the principlesXP emphasizes continuous testing and refactoring as pillars of quality.[7]Black-box testing is used to test software functionality based on input and output without viewing the program code.[12]This method assesses whether the system works according to user requirements. Meanwhile, white-box testing focuses on examining the internal logic of a program, including paths, branches, and code structure, to detect hidden errors. This approach often uses path testing and cyclomatic complexity analysis to ensure the logical integrity of the system.[13].

3. Research Methods

This research was conducted at PT Century Batteries Indonesia (PT CBI) in the spare parts warehouse, focusing on developing a web-based goods expenditure recording feature to improve tracking accuracy and reporting efficiency. The method used was Extreme Programming (XP), a part of Agile Software Development that is iterative and adaptive (Shrivastava et al., 2021). The implementation of XP in this study includes the stages of Planning → CRC Card → Use Case → Activity & Sequence → Class Diagram → Database/UI → Coding → Testing → Release, with a Refactoring and Evaluation process so that the system can be continuously refined based on user feedback.

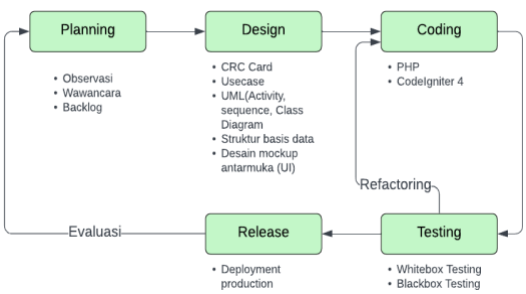

Fig. 1: XP flow that has been adapted to the research context

The Planning phase was conducted through observations and interviews with warehouse staff and the IT team of PT Century Batteries Indonesia (PT CBI) to identify system requirements, such as recording of goods expenditure, stock control, and more structured reporting. The results became the basis for compiling a backlog that was updated in each XP iteration. In the Design phase, the design was carried out starting from CRC Card, Use Case, Activity and Sequence Diagram, to Class Diagram and ERD to support transactions and reporting. The main workflows include recording expenditure, updating stock in the portal, history/reports, and retrieving on-hand data from Infor on a read-only basis.

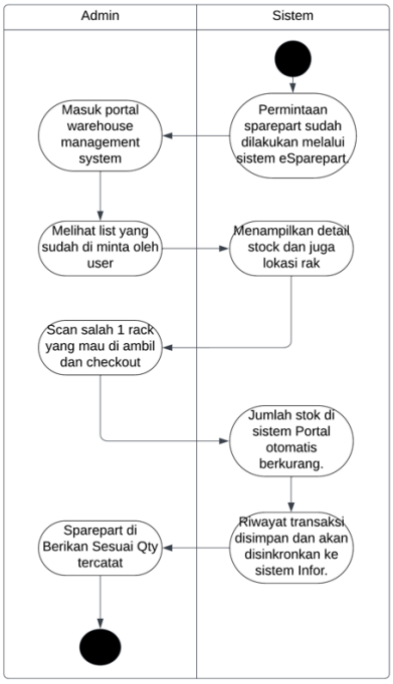

Fig. 2: Flow chartSystem After Web System Implementation

In addition to workflow mapping, the user interface (UI) design focused on ease of use, consistent appearance, and intuitive navigation. The coding phase was conducted using CodeIgniter 4 and Microsoft SQL Server, supporting scanner/PDT input and read-only pull integration from Infor. Development was carried out iteratively according to XP principles. Refactoring was carried out continuously during coding to improve the internal structure without changing functionality, maintain code quality, and ensure the system remains maintainable and stable before being moved to production.

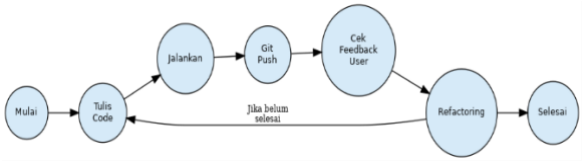

Fig. 3: Refactoring Overview

The Testing phase is carried out to ensure the system functions according to user requirements through two methods: Blackbox testing to test the functionality of key features without viewing the code, and Whitebox testing to verify internal logic using statement and branch coverage. After successful testing, the Release phase is carried out by deploying the system to PT CBI's local server so that warehouse users can conduct real-world testing and provide feedback. Revisions will be carried out iteratively according to XP principles to improve the system's functionality and usability.

4. Result and Discussion

4.1. Planning

In the Planning stage, observations and interviews were conducted at the PT Century Batteries Indonesia (PT CBI) warehouse to identify problems with the manual spare parts system, such as paper recording, data discrepancies between Infor and the Portal, the absence of an audit trail, and difficulties in reporting and finding shelf locations. Based on these findings, a solution was designed in the form of a web-based spare parts management system with QR code checkout features, Infor on-hand data pull (read-only), audit trails, automatic reports, and an interactive shelf map display. Next, a Product Backlog was compiled containing key features such as user management, checkout, Infor integration, history tracking, and expense reports with work priorities based on the system's utility value so that development runs efficiently and in a targeted manner.

4.2. Design

In the design phase of the Extreme Programming method, system requirements are converted into an object-oriented technical model. The process begins with the creation of a CRC (Class Responsibility Collaborator) Card to map responsibilities and collaboration between classes. Table 4 shows the five main classes identified: User handles authentication and session management by collaborating with Auth and AuditTrail; Sparepart stores master data on items and stock; Portal interacts with Transaction; Transaction records each checkout, validates quantities against stock, and displays history with the support of User, Sparepart, and AuditTrail; Report is responsible for generating Excel reports based on Transaction data; while AuditTrail records all critical system activities for audit purposes. This collaboration forms a complete operational flow—from login, item search, checkout with validation, to read-only Infor on-hand synchronization.

Table 1: CRC *Card*

Class	Responsibilities	Collaborators
User	- Log in - Access system features according to role (eg: Warehouse Staff) - Logging user activity	*Auth*, Transaction, AuditTrail
Spare parts	- Save spare part details (name, code, location) - Displays spare parts list Connected to stock data	*Transaction*
Transaction	- Recording stock taking - Record time, amount, and user - Display transaction history	*Spare parts*, User, AuditTrail
Report	- Generate daily/monthly stock reports - Presenting reports on collection activities - Export report to Excel	*Transaction*, User
AuditTrail	- Record any important changes to the system - Save time, user, action	*User*, Transaction

The CRC Card design was then broken down into a Use Case Diagram (Figure 4) that details user interactions with the system. The diagram displays two separate systems: Portal for checkout recording and Infor as an on-hand (read-only) reference source. Warehouse staff can log in, search data, view Portal stock, checkout with validation, view history, export reports, look up shelf addresses, and perform on-hand withdrawals from Infor. Integration is one-way—Portal only reads on-hand values without pushing them to Infor.

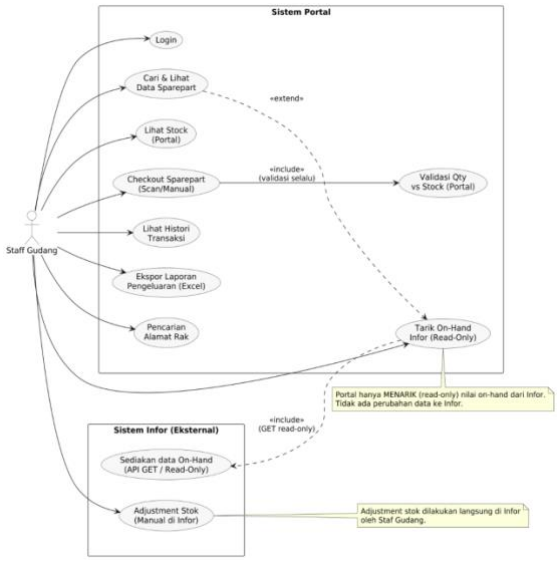

Fig. 4: Sparepart Portal Use Case Diagram

*Activity Diagram*This diagram illustrates the flow of activities between users and the system in the warehouse operational process, divided into two swimlanes (User and System). This diagram displays the main processes such as login, displaying the order list, spare part checkout with stock validation, shelf address search, synchronization of on-hand data from Infor (read-only), transaction history browsing, and report export to Excel. Each flow shows valid/invalid decisions and API interactions that support data synchronization between the Portal and Infor. The following is an activity diagram of the spare part checkout process.

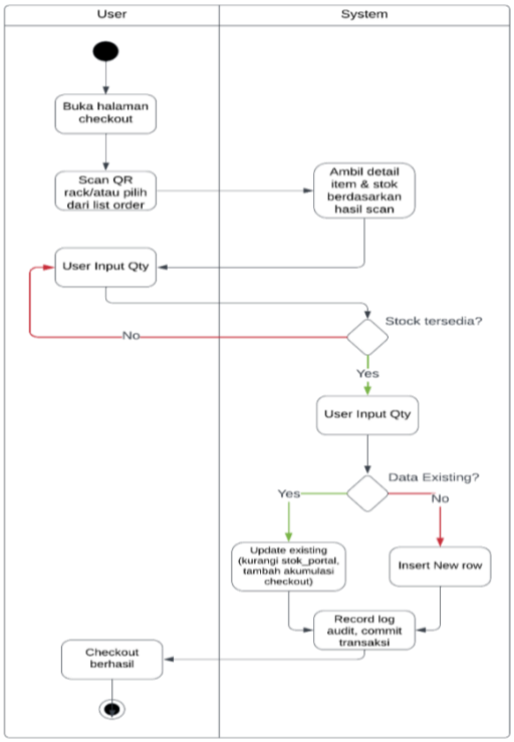

Fig. 5: Activity Diagram of the Sparepart Checkout Process

This diagram depicts the class structure in the system along with its attributes and relationships.

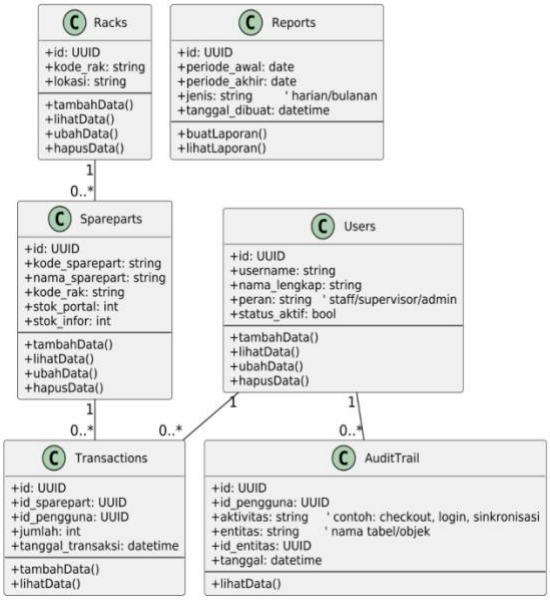

Fig. 6: *System Class Diagram* WMS

The Database Structure section describes the design and mapping of the object model to physical tables to support a web-based warehouse management system. The database design focuses on integrity, efficiency, and ease of tracking activity. Primary classes such as User, Sparepart, and Transaction are mapped to the users, sparepart_storages, and sparepart_checkout tables, respectively, while temporary entities such as Cart are represented by sparepart_checkout_temp. Master and transaction data are separated for performance and auditability, while integration with Infor is read-only with no redundant storage. This structure enables efficient data synchronization, user-verified transaction recording, and supports automated report export without data duplication.

The user interface design was based on input from warehouse users at PT Century Batteries Indonesia (PT CBI). The initial design was created using an interface design application (mockup) to visualize the menu layout, buttons, and user interaction flow before implementation.

Each design is designed to support user-friendliness, visual consistency, and warehouse workflow efficiency. These include login pages, order lists, addressing, checkout, history/adjustment summaries, and report exports.

Below is a mockup of the checkout page on a PDT scanner (mobile) as part of the system interface design.

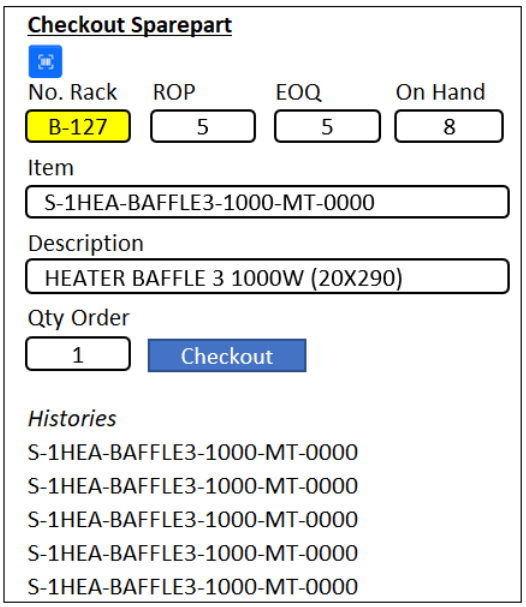

Fig. 7: Design *Mockup* page *checkout* on *scanner* PDT (*mobile*)

4.3. Coding (System Implementation)

The spare parts management application was developed with CodeIgniter 4 (CI4) based on MVC and the WarehouseManagementDB database, integrated read-only with the Infor system for stock data retrieval. The system is equipped with server-side validation, CSRF protection, and audit logs to maintain data security and integrity. Key features include secure login, checkout with stock and database transaction validation, shelf location search, on-hand synchronization, history reconciliation, and Excel report export. Refactoring added an adjustment time column to improve traceability without changing the read-only nature to Infor. The following is a code snippet for login with session management.

```
// AuthController
// Verifikasi kredensial
$user = $this->userModel->where('username', $request['username'])->first();
if ($user && password_verify($request['password'], $user['password'])) {
    // Local login successful
    $sessionData = [
        'id' => $user['id'],
        'name' => $user['name'],
        'email' => $user['email'],
        'role' => $user['role'],
        'npk' => $user['npk'] ?? null,
        'type_user' => 'local',
        'isLoggedIn' => true
    ];
    // Regenerasi session ID untuk mencegah session fixation
    session()->regenerate(true);
    // Set session data
    $this->session->set($sessionData);
    // Role-based redirect
    if ($user['role'] === 'sparepart') {
        return redirect()->to(base_url('sparepart'));
    }
    if ($user['role'] === 'incoming') {
        return redirect()->to(base_url('incoming'));
    }
    return redirect()->to(base_url('/'));
}
```

Fig. 8: Login Code Snippet with Session Management

This code shows the hash-based authentication process: the system looks up the user, verifies password_verify, regenerates the session ID, saves the user attributes and roles to the session, and then performs a role-based redirect (sparepart/incoming/home).

```php
// Sparepart\PreparationController::checkoutAction()
public function checkoutAction() {
    $data = $this->request->getPost('data');
    $qty = (int) $data['qty'] ?? 0;
    $currentTime = date('Y-m-d H:i:s');
    $existing = $this->checkoutModel->getStockByBack(
                $data['location_id'] ?? null);

    try {
        $this->db->transBegin();

        if ($existing) {
            $onhand = (int) $existing['qty_onhand_portal'];
            $checkedOut = (int) $existing['qty_checkout'];

            // Validasi stok Portal
            if ($qty > $onhand) {
                return $this->response->setJSON([
                    'status' => 400,
                    'message' => 'Stock dari portal tidak mencukupi.'
                ]);
            }

            // Update stok & audit time
            $this->db->table('sparepart_checkout')
                ->where('id', $existing['id'])
                ->update([
                    'qty_onhand_portal' => $onhand - $qty,
                    'qty_checkout' => $checkedOut + $qty,
                    'updated_at' => $currentTime,
                ]);
        }

        // Jejak temp (bila ada cart)
        if (!empty($data['id_cart'])) {
            $this->checkoutTemplate1->saveCheckoutTemp([
                'id_cart' => $data['id_cart'],
                'time_checkout' => $currentTime,
            ]);
        }

        $this->db->transCommit();
        return $this->response->setJSON([
            'status' => 200,
            'message' => 'Checkout successful.'
        ]);

    } catch (\Throwable $e) {
        $this->db->transRollback();
        return $this->response->setJSON([
            'status' => 500,
            'message' => 'Checkout error.'
        ]);
    }
}
```

Fig. 9: Transaction Code Snippet *Checkout* & Validation

Code This demonstrates the implementation of a spare parts checkout with strict quantity validation, using database transactions to ensure data consistency. The system validates positive input, checks Portal stock availability, and performs automatic rollbacks in case of errors to maintain data integrity.

*Refactoring*This was done to add adjustment completion time recording to each checkout transaction for more accurate tracking. The main change is on the backend, namely the addition of logic to mark transactions that have been adjusted in Infor. The following code snippet demonstrates the implementation of this function in the controller.

```php
public function markAdjustmentNone()
{
    $cartId = $this->request->getPost('cart_id');
    $userRole = session()->get('role');
    // Verifikasi hak akses (contoh kebijakan peran)
    if (!in_array($userRole, ['admin', 'sparepart'])) {
        return $this->response->setJSON([
        'status' => 403,
        'message' => 'Unauthorized']);
    }
    try {
        $currentTime = date('Y-m-d H:i:s');
        // Update time_adjustment (idempotent)
        $this->db->table('sparepart_checkout_temp')
            ->where('id_cart', $cartId)
            ->update(['time_adjustment' => $currentTime]);
        // Audit trail
        log_message('info', "Adjustment completed for cart ($cartId)");
        return $this->response->setJSON(['status' => 200,
            'time_adjustment' => $currentTime,
        ]);
    } catch (\Exception $e) {
        return $this->response->setJSON([
        'status' => 500,
        'message' => 'Failed']);
    }
}
```

Fig. 10: Mark Adjustment Complete" Controller Code Snippet.

4.4. System Testing

Blackbox testing is used to ensure core functionality works as required without looking at the code. Coverage: login/authorization, order list, addressing, view stock (Portal), pull on-hand (Infor, read-only), checkout, history/adjustment summary, and (optional) Excel export. Test environments: main branch (last commit), accounts with different roles (sparepart & non-sparepart), and normal and degraded network conditions (Infor API timeout simulation). Pass criteria: correct status/payload, appropriate UI/messages, and correct data changes.

Table 2: Black Box Test Scenario & Results

ID	Feature	Scenario & Input	Results
BB01	*Login*	Valid credentials	*Redirect*according to role; create a new session
BB02	*Login*	Incorrect credentials	Error message displayed; did not create session
BB03	Authorization	*User*non-*spare parts*spare part module access	Rejected/redirected; content not showing
BB04	*Order List*	GET date of data existence/absence	data displayed / totalData=0 + UI "not found"
BB05	*Addressing*	Search shelf valid / not found	Item table appears / message "address not available"
BB06	View stock	Open item details (including stock 0)	Stock value displayed; stock 0 is marked
BB07	*Pull On-Hand*	Normal / timeout when clicking Sync	Updated references + timestamp / error displayed; local data unchanged
BB08	*Checkout*	Valid Qty (>0 & ≤ Portal stock)	Transaction saved; Portal stock reduced; audit recorded
BB09	*Checkout*	Qty > Portal stock	Rejected by validation; stock remains
BB10	*Checkout*	*Race condition*(2 requests almost simultaneously)	One rollback; final data is consistent
BB11	History/ Adj.	There is data / empty	*list*show / empty state
BB12	Export	Valid / invalid period	.xlsx downloaded / error displayed

All core cases pass. Validation prevents quantity from exceeding Portal stock; race condition scenario resolved (rollback of one transaction); Infor API failure is secure (local data unchanged, message clear).

4.5. Release

The system was released on PT Century Batteries Indonesia's (PT CBI) internal server after passing black box and white box testing. Deployment was carried out in collaboration with the IT team with a SQL Server database configuration according to the architecture in Chapter 3. Connection verification to the Infor system for the on-hand recall feature was performed to ensure smooth communication between the systems.

*Quick test*The process included login, checkout (valid/invalid scenarios), and on-hand synchronization—all successful without any critical errors. A short introductory session was provided to warehouse staff to familiarize them with the workflow: login, checkout (QR scan/manual), history, and report export.

*Monitoring*Three days later, the system was stable without any critical errors. User feedback was positive, particularly regarding the ease of checkout and the Portal-Infor reconciliation feature. Minor suggestions (loading optimization, UI search) were recorded as part of the next iteration's backlog. In line with XP principles, feedback informs ongoing development.

4.6. User Acceptance Test (UAT)

System acceptance validation was conducted through informal discussions with warehouse users. They were asked to use the system for daily transactions and provide a simple rating (scale of 1-5) and direct feedback.
Questionnaire Items:

1. Recording requests/expenses in a computerized manner (checkout recording is running and stored)
2. Validate outgoing quantity against Portal stock, with Infor on-hand reference (read-only)
3. Transaction history (who/what/when/how much) can be reviewed for auditing.
4. Reports can be exported to Excel by period
5. Portal vs. Infor on-hand stock comparison information is available for verification/reconciliation.
6. User identity when the transaction is saved (recorded role/ID)
7. *Dashboard*spare part stock status (low priority) — not yet released → N/A

Table 3: User Acceptance Test

Aspect	Evaluation	Comment
Checkout quantity recording	**4.8/5**	Easy to use
Portal qty vs stock validation (+ Infor on-hand reference)	**4.7/5**	Good at minimizing errors
Transaction history (who/what/when/how much)	**4.1/5**	For the purposes of a complete audit, there is no UI yet
Export Excel expenses	**4.3/5**	—
Comparison of Portal vs. on-hand Infor for adjustment verification	**4.9/5**	Proven to avoid discrepancies
User identity is recorded in the transaction	**3.5/5**	There is no UI for certain pages, you have to look via log.
*Dashboard*stock status (low priority)	**—**	—

Based on the questionnaire and user field testing, the application is accepted by users. This UAT complements the blackbox: the blackbox checks for compliance with specifications, while the UAT assesses usability in the field and confirms that user needs are met.

5. Conclusion

A Sparepart Warehouse Portal was successfully developed using Extreme Programming (XP) methods to address the manual recording of goods releases at PT Century Batteries Indonesia (PT CBI). The system was designed as an independent portal that complements the main system (Infor), focusing on checkout recording, stock validation, transaction history, and reporting without changing the restricted access system.

The system design was translated from the product backlog observation results using an object-oriented approach with CRC Card and UML (Use Case, Activity, Class Diagram). The main features implemented include: (1) checkout recording with quantity validation against Portal stock; (2) on-hand pull from Infor read-only for reconciliation; (3) transaction history with user and time metadata; and (4) Excel report export. The operational flow of "checkout in Portal → manual adjustment in Infor → on-hand pull" has been proven to reduce data discrepancies that previously arose due to manual recording.

Stock consistency mechanism is maintained through Adjustment Summary which provides real-time comparison of Portal vs. on-hand Infor stock, facilitating reconciliation without the need for write access to the main system. System evaluation shows User Acceptance Test (UAT) with an average of 4.4/5 from 6 warehouse users, with the highest scores in the Portal-Infor Comparison feature (4.9/5) and Checkout Recording (4.8/5). Black-box testing proves that all functional scenarios run as expected, while white-box testing uses the path basis technique on 8 critical functions (login, checkout, syncStock, etc.) with low cyclomatic complexity (CC=2-4), ensuring all logic paths are executed and code maintainability is maintained iteratively with small releases and continuous feedback, in line with XP principles. For the next iteration, it is recommended to add structured audit tables to improve traceability, implement caching on reference data for performance optimization, complete the stock monitoring dashboard (F07), integrate Single Sign-On (SSO) or two-factor authentication for security, and implement a simple deployment pipeline with scheduled backups to speed up the release cycle and reduce human error.

References

[1] A. Herdiansah, R.I. Borman, and S. Maylinda, "A Laravel Web Framework-Based Laminating Process Quality Control Monitoring and Reporting Information System," Jurnal Tekno Kompak, vol. 15, no. 2, pp. 13–24, Aug. 2021, doi: 10.33365/jtk.v15i2.1091.

[2] N. DeHoratius and A. Raman, "Inventory record inaccuracy: An empirical analysis," Management Science, vol. 54, no. 4, pp. 627–641, 2008, doi: 10.1287/mnsc.1070.0789.

[3] Y. Zhang and F. Pan, “Design and Implementation of a New Intelligent Warehouse Management System Based on MySQL Database Technology,” Informatica, vol. 46, no. 3, pp. 355–364, 2022, doi: 10.31449/inf.v46i3.3968.
[4] S. Chen, W. Meng, W. Xu, Z. Liu, J. Liu, and F. Wu, “A Warehouse Management System with UAV Based on Digital Twin and 5GTechnologies,” in 2020 7th International Conference on Information, Cybernetics, and Computational Social Systems, ICCSS 2020, 2020, pp. 864–869. doi: 10.1109/ICCCS52145.2020.9336832.
[5] AA Ali, “IoT Based Warehouse Management System Leveraging On RFID and Cloud Platform Technologies,” inATNT 2024, 2024. doi: 10.1109/ATNT61688.2024.10719252.
[6] S. Gawande, R. Agrawal, R. Ingole, P. Akotkar, and AK Shahade, “Review of Warehouse Management Systems,”Int. J. Adv. Res. Sci. Commun. Technol., pp. 53–56, 2023, doi: 10.48175/ijarsct-9541.
[7] K. Beck and C. Andres, Extreme Programming Explained: Embrace Change, 2nd ed. Boston, MA, USA: Addison-Wesley, 2004. ISBN 978-0-321-27865-4.
[8] A. Shrivastava, I. Jaggi, N. Katoch, D. Gupta, and S. Gupta, “A Systematic Review on Extreme Programming,”J. Phys. Conf. Ser., vol. 1969, no. 1, 2021, doi: 10.1088/1742-6596/1969/1/012046.
[9] HR, I.R., T. Alam S., and H., “Optimizing Warehouse Operations in Bangladesh: Leveraging IoT and Cloud Migration for Enhanced Security and Efficiency,” inFiCloud 2024, 2024. doi: 10.1109/FiCloud62933.2024.00061.
[10] KR Elmasri and SB Navathe, Fundamentals of Database Systems, 7th ed. Pearson, 2016.
[11] G. Kurniawan, F. Adnan, JA Putra, U. Jember, and P. Correspondence, “USER INTERFACE AND USER EXPERIENCE DESIGN OF BATIK CLOTH E-COMMERCE APPLICATION IN REZTIS BATIK UMKM USING DESIGN THINKING APPROACH,” vol. 10, no. 3, pp. 551–560, 2023, doi: 10.25126/jtiik.2023106733.
[12] JB Ardika and N. Ratama, “Implementation of QR Code Using Google Application Programming Interface (API) in Building a Warehouse Information System with Web-Based Extreme Programming Method (Case Study: PT Bell Flavors & Fragrances Indonesia),”J. Technol. Sis. Inf. And App., vol. 5, no. 1, p. 47, 2022, doi: 10.32493/jtsi.v5i1.16165.
[13] R. Andrian Ibrahim and G. Saktian Laksito, "Optimization of White Box Testing by Utilizing Branching and Repeating Structures in Java Programs Using Base Path,"Int. J. Math. Stat. Comput., vol. 2, no. 2, pp. 85–89, 2024.