

National Application for Tracking and Recovering Lost Official IDs

Amenah Abdullah^{1*}, Wael Shalan Abd Alaziz²

¹Department of Computer Science, College of Computer Science and Information Technology,
University of Sumer, Thi-Qar, Iraq

*Corresponding Author: amnaabd999@gmail.com

Phone number: 07810955172

Abstract

Loss of official documents (passport, national identity, voter card, etc.) are frequent problems in Iraq without the existence of a national system to track and recover these important lost items. In this research, we provide a practical solution to this problem through mobile applications and with the help of Internet of Things (IoT) network services. Google's cloud database (Firebase) has been relied upon, where records are filled on the cloud, by reading the icons in the lost items, using short-term communication technologies such as; Near Field Communication (NFC), Bluetooth and Radio Frequency Identification (RFID). The system provides a mechanism for recording, detecting and alerting missing documents found and locating automatically located them in record time. The waiting system has been activated in the event of an interruption of the connection to the Internet and the postponement of operations until the return of the call, in addition to the formation of secret codes consisting of six numbers for the purpose of receiving the lost items. In this application, we present an initial practical model for an Android application consisting of about (16,500) Kotlin-language code, spread over (78) programming files.

Keywords: Internet of Things; Smart Locker; Near Field Communication; Cloud Computing; Mobile Application; Edge Computing.

التطبيق الوطني لتتبع واستعادة الهوية الرسمية المفقودة

آمنة عبدالله^{1*}, وائل شعلان عبد العزيز²

^{1, 2} قسم علوم الحاسوب، كلية علوم الحاسوب وتكنولوجيا المعلومات، جامعة سومر، ذي قار، العراق

الخلاصة

فقدان المستندات الرسمية (جواز السفر، الهوية الوطنية، بطاقة الناخب، إلخ) هي مشاكل متكررة في العراق دون وجود نظام وطني لتتبع هذه العناصر المفقودة المهمة واستعادتها. في هذا البحث، نقدم حلاً عملياً لهذه المشكلة من خلال تطبيق الهاتف المحمول وبمساعدة خدمات شبكة إنترنت الأشياء. تم الاعتماد على قاعدة بيانات السحابة من Google، حيث يتم ملء السجلات على السحابة، من خلال قراءة الرموز الموجودة في العناصر المفقودة، باستخدام تقنيات الاتصال قصيرة المدى. يوفر النظام آلية لتسجيل واكتشاف وتنبيه المستندات المفقودة التي تم العثور عليها وتحديد موقعها تلقائياً وتلقائياً في وقت قياسي. تم تفعيل نظام الانتظار في حالة انقطاع الاتصال بالإنترنت وتأجيل العمليات حتى عودة المكالمة، بالإضافة إلى تكوين رموز سرية تتكون من ستة أرقام لغرض استلام البنود المفقودة. في هذا التطبيق، نقدم نموذجاً عملياً أولياً لتطبيق Android يتكون من حوالي (16500) كود لغة Kotlin، موزعة على (78) ملف برمجة.

1. Introduction

Considering the tremendous development of electronic services and the rapid transformation of the world into e-governance, it is necessary to take advantage of all the services and capabilities provided by this transformation, and to integrate the Internet of Things with this transformation provides ease and intelligence for these services. Since the printed national card, passport and other official documents are among the most important documents, it is necessary to protect and recover them using modern technologies. This research highlights a proposed national system for registering, tracking, and locating lost official documents, with the participation of Iraqi citizens.

With the rapid evolution of the Internet of Things (IoT), cloud computing, mobile applications and smart wireless communication systems, companies and researchers are starting to introduce smart tracking, monitoring and recovery solutions for important or precious lost things [1], [2]. As is known, the IoT system is generally defined as an integrated network of smart devices capable of collecting, exchanging and processing data using Internet infrastructure and cloud services. Recently, these technologies have been widely used in smart cities, smart logistics, smart lockers, healthcare systems and secure asset management applications [3].

Near Field Communications (NFC) technology is one of the most important technologies used in short-range wireless communication and modern IoT, due to its simplicity, low power consumption, secure communication compatibilities and compatibility with smart devices such as smartphones and computers [4]. NFC technology has been used in contactless payment applications, smart access systems, authentication systems, lost-item locator systems, and more applications [5].

Several studies have attempted to use NFC Tags to find and manage documents and lost items, the researchers in [6] proposed the NFC-LMS system with the Lost Articles mobile application. While several researchers focused on smart locks or smart and secure storage, for example, researchers in [7] developed a sure storage system based on IoT services, using low-energy Bluetooth and cloud computing technologies. In a similar way, the researchers in [8] hired Radio Frequency Identification (RFID) technologies for object identification and tracking applications. Moreover, IoT technologies have been extensively investigated for communication, monitoring and management purposes [9,10]. These technologies constitute the technical basis for developing intelligent document recovery systems.

Besides NFC and RFID technologies, Bluetooth Low Energy (BLE), cloud computing, edge computing and mobile applications have become important technologies in modern IoT infrastructures. These technologies provide secure communication, centralized data management, temporary local processing capabilities and real-time interaction between users and smart devices. Therefore, integrating these technologies enables the development of smart IoT based systems capable of managing sensitive documents and supporting secure recovery procedures.

In addition to academic studies, many industrial solutions have shown the practical importance of smart tracking and retrieval techniques. RFID-based document tracking

systems have been employed for managing sensitive documents and organizational records, whereas smart retrieval platforms such as Amazon Locker have demonstrated the feasibility of combining cloud services, electronic locking systems, authentication mechanisms and automated notification services within large-scale environments [11],[12]. However, these systems primarily aim to deliver parcels and inventory management rather than retrieve official governmental documents. Table I summarizes the differences between existing approaches and the proposed system.

Table 1- Comparison of the current methods and the proposed system.

Feature	Lost item Systems [6]	Smart Lockers [7],[8]	RFID Tracking Systems [11]	Amazon Locker [12]	Proposed System
NFC Identification	✓	x	x	x	✓
RFID Tracking	x	x	✓	x	Optional
Cloud Services	✓	✓	✓	✓	✓
Mobile app.	✓	✓	✓	✓	✓
Lost-item recovery	✓	x	Partial	x	✓
Secure storage	x	✓	✓	x	✓
Official Document Support	x	x	Partial	✓	✓
SMS and E-mail Notifications	x	x	x	✓	✓
Authentication Code Generation	x	x	x	x	✓
Offline Synchronization	x	x	x	x	✓
National-Level Framework	x	x	x	x	✓

From the above discussion, previous studies focused mainly on the management of missing items, secure storage systems, or RFID-based tracking techniques.

However, they did not specifically address the problem of recovering official government documents. Motivated by these developments, this paper proposes a national NFC-based application to track and recover lost official identifiers using NFC, cloud computing and mobile applications. The proposed system integrates many technologies into a unified framework, including NFC-based definition, cloud database services, automatic owner notifications, secure recovery procedures, and offline temporary operational capabilities.

The originality of this work is the result of the integration of these technologies; NFC-based lost-item systems [6], Bluetooth based smart storage systems [7], RFID identification technologies [8], IoT foundations [1], [9], and smart retrieval approach inspired by Amazon-Locker into an integrated framework designed specifically for recovering official documents. To the best of the authors' knowledge, no previous work has proposed a citizen-oriented national platform for tracking and recovering lost governmental IDs and sensitive documents. In addition, the proposed system introduces a waiting mechanism during Internet interruptions and employs a six-digit authentication code to ensure secure document delivery.

Although the proposed system provides a practical and low-cost solution, several limitations still exist. The current implementation depends on NFC-enabled smartphones and Internet connectivity for synchronization. In addition, the developed prototype has not yet been deployed on a national scale. Future work may include integrating long-range RFID technologies, artificial intelligence techniques, GPS-supported smart recovery stations, and advanced cybersecurity mechanisms suitable for large-scale governmental deployment.

The remainder of this paper is organized as follows. Section 1 reviews related works and background technologies. Section 2 presents the proposed system architecture and its hardware and software components. Section 3 describes System Design and Implementation. Section 4 presents experimental results and system evaluation. Finally, Section 5 concludes the paper and discusses future research directions.

2. System Architecture

The proposed system is organized as a layered software system with three coordinated layers: an application layer, a cloud layer, and a communication layer. The application layer is a native Android application that serves two audiences through separate interfaces, one for the citizens who own the documents and one for the administrators who operate the service. Google Firebase has been deployed as the cloud layer on the project, and therefore it makes all users directly connected to a database, secure method, and auto-sync of clients data. The communication layer defines how data moves between the application and the cloud and how each client is authenticated and authorized. The overall architecture of the proposed system and the interaction between its three main layers are illustrated in Fig. 1.

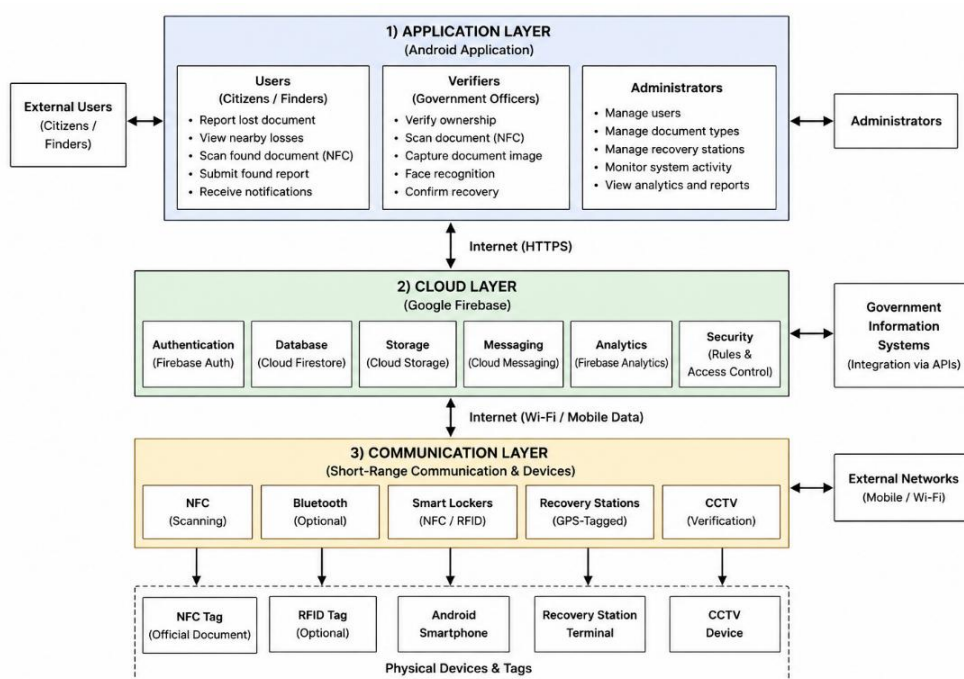


Figure -1 Overall architecture of the proposed system.

2.1 Cloud Layer

The cloud layer uses Google Firebase as a managed backend. This means we do not have to run and maintain a custom server. Cloud Firestore [15] stores the authoritative state in nine collections, summarized in Table 2. Firebase Authentication issues verify the credentials of citizens and administrators. Cloud Functions implement the event driven business logic, such as sending email, fanning out push notifications, and provisioning administrator accounts. Firebase Cloud Messaging delivers push notifications to citizen and administrator devices, and Cloud Storage keeps the profile and document images [16].

Table 2- Summary of the Firestore collections used by the proposed app.

Collection	Purpose
users	Citizen profiles, including a human-readable unique identifier (e.g., SDB-2026000001).
admins	Administrator accounts with role (SUPER_ADMIN or ASSISTANT_ADMIN) and active status.
slots	Per-slot state: status, current tag and owner, secret code, deposit/pickup timestamps, failed attempts.
tags	Registered documents bound to NFC UUIDs, including pending tags awaiting NFC linking.

notifications	Per-user and per-admin notifications; act as triggers for Cloud Functions to send email and FCM.
lostReports	User-submitted lost-document reports with status workflow (pending → in-review → found/closed).
branches	Physical service locations (branches) with geo-coordinates, working hours, and slot capacity.
transactionLogs	Append-only audit log of all deposits, pickups, manual unlocks, and administrative actions.
systemSettings	Global tunables: total slots, long-stay threshold, failed-attempts limit, support contacts.

2.2 Application Layer

The Android application is written natively in Kotlin [17] with a Jetpack Compose [18] user interface and follows the Model View Model (MVVM) pattern with a unidirectional data flow. A single App View Model exposes the reactive state of the whole application through State Flow, and it is fed by a Unified Repository that is the only gateway to data. The repository consolidates two sources behind one interface: the Firebase managers for network access and a local Room database for the on-device cache. The interface is right to left first to match the Arabic speaking users. Two separate navigation graphs keep the two experiences apart, one for the citizen (home, my documents, pickup location, lost report, pickup, profile, and settings) and one for the administrator (slot dashboard, users, tags, deposit simulator, logs, branches, and settings).

2.3 Communication Layer

All clients communicate with Firestore through real-time snapshot listeners, which deliver state changes to every client within a small, bounded delay. When it comes to writing data there are three ways this is done. Some updates can be written directly. This is the case for low conflict updates. However, for operations like pickup, where everything needs to be consistent and happen at the same time these are written inside atomic Firestore transactions. If the device is offline when a write is issued it is placed in a queue. Then when the device is online this write is replayed. Cloud Functions are like engines that handle tasks. For example, when a document is created a function triggered by sending a matching email and push notification. This all happens without stopping the client that created the document from doing things. The cloud layer uses Google Firebase. This is very useful for the cloud layer.

3. System Design and Implementation

3.1 Data Model

The data model is about the slot and what state it is in. A slot can be in one of five states. It can be EMPTY, which means it is available. It can be OCCUPIED, which means it has a document in it that we know about. The slot can also be in a state called LONG_STAY which happens when a document has been in the slot for long and an administrator needs to review it. Sometimes a slot can be DISABLED, which means it is locked and cannot be used usually because someone tried to pick up a document many times. It needs to be fixed. Then there is the UNKNOWN_TAG state, which is when a document is in the slot, but its NFC tag is not in our list of tags. A document is connected to one person and one type of document like an (ID driving license or passport. It can also be a residence card, voter card, student card, health care, bank card or some other type of card. Sometimes a document can exist without being connected to a chip and it will just be waiting for an administrator to link it. We also have lists for notifications, lost reports and transaction logs, which we use to send messages and keep track of what's happening [22]. These lists are mostly used for writing, which means we add information to them all the time.

3.2 Deposit Workflow

When someone finds a document, they use the application to let us know. The person using the application must give us the NFC identifier of the document. Then the application checks the tags we have on file. If we find a match, we know who the document belongs to. If we do not find a match [19], we will make a note of it so we can investigate it later. Next the application looks for the available spot to store the document. It comes up with a code that is six digits long. Then it makes three changes at the time: it marks the spot as taken it sends a message to the owner that we found their document or it sends a message to the people in charge if we do not know who the document belongs to and it adds a new entry, to the list of things that have happened. When we send this message it triggers something that sends an email to the owner with a QR code. It also sends a notification to the owner's device [16] to let them know that we found their document the Found document. That we have it. The application is used for the Found document.

3.3 Pickup Workflow and Two Factor Verification

The pickup is an important part of the system when it comes to security. That is why the pickup is done as a Firestore transaction [15]. The pickup is also controlled by two things. The owner will open the application, read the notification, and will start the pickup. The first factor is the six digit secret code, which is stored only in the slot record and is shown to the owner only after the owner has signed in. The second factor is a face check that runs entirely on the device. The owner enrolls a face once, and the application stores a numeric face descriptor, an embedding of 192 values, on the owner's user document; the face image itself is never stored, so it cannot be reconstructed from the descriptor. At pick up the application captures the live face, computes its embedding, and compares it with the stored embedding using cosine similarity, accepting the match when the similarity is at least a fixed threshold.

Only when the code is correct and the face matches does the application run the pickup transaction [13],[14]. The transaction reads the slot record, rejects it if the state is DISABLED, and compares the submitted code with the stored one and when the pickup is successful the slot will be reset to empty. All the information about the owner will be cleared up. A new log entry will be added. If the wrong code is used the number of failed attempts will go up. When this number reaches the limit, which's three by default the slot will be set to disabled. An alert will be sent to the administrator. The pickup is done in a way that two people cannot successfully pick up the slot at the same time. The system also offers a signed QR code. This QR code has the information as the code. It is given as an option, for people who do not want to type the code by hand. The pickup is a security operation of the system [20].

3.4 Offline Synchronization

The app starts in offline mode. It has a local database called Room that stores information from nine tables, which are copies of the collections in Firebase [15], plus an extra table for operations that are waiting to happen. When the user tries to implement a change or an inquiry in the application, and if the device is offline, it saves these modifications as a pending process. These operations include a JSON payload, the type of operation and there are 18 types of operations like, creating, updating, deleting, uploading an image or files, changing a notification, and its status. The app checks for internet connectivity periodically to clear the queue of pending operations when the device gets back online. If an operation fails, it will try again up to three times before giving up and letting the user know something went wrong. In fact, this flexibility between the application and the connection status is very important to always update the local status until the device and the cloud database are connected, which, of course, occurs in the background of the application [18].

3.5 Security Model

The system has two main users: regular people and those in charge. The proposed system is used by two types of users: ordinary users and application managers. For successful access, the application needs the user to enter the account identifier and the password to enter via Firebase Authentication. Then there are security rules, called Firestore Security Rules, which only allow everyone to see and change their data. They can look at their user information, tags and messages, and they can update their profile, and a private field called "Pickup" is protected with a code in case any of the official documents of the user are lost[20]. The people responsible for the application have different levels of power. The top administrator can add or remove other administrators using special cloud functions called build and update AdminAccount. According to these rules, the assistant administrators account enables them to perform and track daily operations, and this is done automatically when creating an administrator account by creating a file that stores all the powers and permissions of the administrator account automatically, and this is due to the use of the AdminClaims function. These security rules use application user information to determine permitted actions. The file storage system, called CloudStorage, uses the same mechanism to verify user identity. When

someone uploads an image or file, the system checks the file type and size to ensure it is authorized. In this way, data is secure and access is restricted to authorized users [21].

However, the current security model still has some limitations, particularly its reliance on Firebase authentication and the lack of comprehensive security testing during large-scale deployments. In future studies we might address more sophisticated cybersecurity mechanisms, as well as a more in-depth evaluation of biometric authentication.

3.6 Mobile Application

The mobile application of this project was designed and implemented using Kotlin. This application includes about 16,500 lines of code distributed over 78 files. Jetpack Compose [18] was used for the purpose of creating the GUI, in addition to using MVVM with Single App View Model to give the application an organized look. The connection with Firebase and The Room Cache database was via the Unified Repository Class [17]. As for notifications, the application has three notifications' channels that are executed by Foreground Services. More, there are users' preferences, such as choosing the application language, theme, and the type of security lock, and this is stored using the DataStore function. The application also provides a very important feature, which is On-Device Face Recognition, which combines Google ML Kit and MobileFaceNet, which works with TensorFlow Lite to conduct biometric verification. Having such a feature is important because the client is not required to send data to the server over the Internet for the purpose of verification every time the application is used. In general, this application is designed to run on Android 7 or later in a highly efficient, safe and easy-to-use.

3.7 Application Interface

The application has two main interfaces: one for the user (citizen) and the other for the application administrator. The user interface includes 16 windows to provide services such as account creation and document uploading. Upon registration as a new user, a unique number is automatically generated, such as SDB-2026-000001. This number allows the user to add documents they wish to track and retrieve if lost, simply by adding an NFC-tag and uploading this information to the application. In the event of the loss, a special passcode is generated for the purpose of receiving the lost items in a secure manner, as well as a message containing the details of the place in which the missing items are located, and they are sent to the person concerned in the form of an e-mail and SMS. The process of notifying the lost items is carried out either through the NFC tag enchantment of the document at one of the service points or through the application.

On the other hand, there are 19 windows for the admins and system supervisors. These windows show, for example, the number of lost documents, found documents, and the number of registered documents in the system. In addition, the system allows the administrator to manually intervene in case of software or hardware failure, as it can open the storage boxes manually, modify the data, and the status of the lost items manually as well. In general, the application is designed with high quality for the management of lost items and

user information, especially the NFC tag for each official document that has been added to the application. All screens are rendered with Jetpack Compose and laid out right-to-left to match the Arabic-speaking target audience.



Fig. 2. The citizen interface.

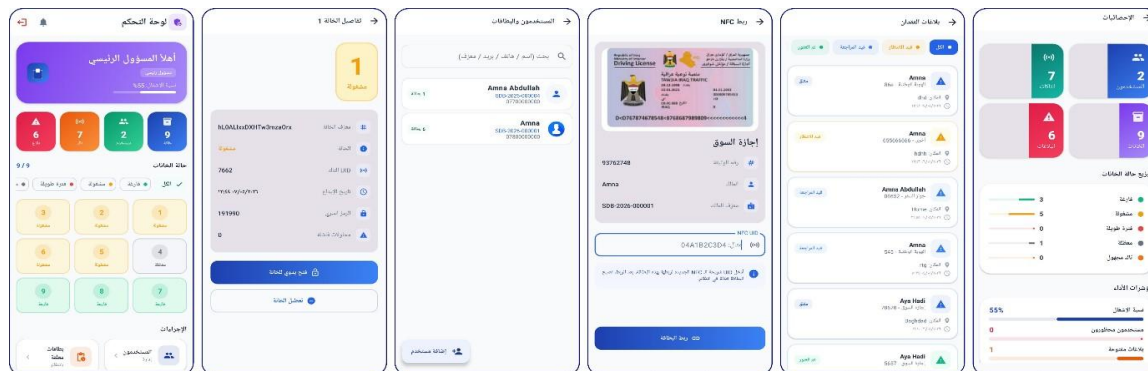


Figure -3 The administrator interface.

3.8 Cloud Functions

The Cloud Functions are implemented in Node.js and consist of eight functions across three areas. The messaging functions (sendCardFoundEmail, sendWelcomeEmail, sendReportUpdateEmail, and sendFcmNotification) are triggered by the creation or the update of documents in the notifications, users, and lostReports collections. The administration functions (deleteAuthUser, createAdminAccount, and updateAdminAccount) are HTTPS callable and are reserved to the super administrator. The identity function (setAdminClaims) runs whenever an administrator document is written and keeps the Firebase Authentication custom claims in sync with the admins collection. Every email is rendered from an HTML template with a server generated QR code [16].

4. Evaluation and Discussion

4.1 Functional Validation

The application was validated end to end against the deployed Firebase backend on a real device. Eight scenarios were exercised: (i) an administrator creates a new citizen account,

including the automatic generation of a unique human readable identifier through a Firestore transaction on the systemSettings document; (ii) a citizen creates a tag in the pending state and an administrator later links it to an NFC identifier; (iii) a deposit is recorded through the administrator simulator; (iv) a pickup succeeds with the correct secret code and a matching face; (v) the failed attempts lockout triggers after three wrong codes, moving the slot to DISABLED and creating an administrator alert; (vi) an administrator performs a manual unlock; (vii) a citizen submits a lost report and an administrator updates its status; and (viii) offline behaviour is checked by toggling airplane mode during writes and confirming both the deferred queueing and the synchronization after reconnection. And here all the scenarios are successfully completed, and the Firestore remains in a constant state under concurrent operations.

The testing environment consisted of a developed Android application running on a real device connected to an installed Firebase server. Each scenario was executed individually, and the resulting application response and corresponding changes to the Firestore database were examined to confirm the successful execution of the tested operation.

4.2 System Parameters

Table 3 shows the numbers that the application uses. These numbers are always the same because they come from the application itself. The internet connection or the device you use do not change these numbers, so you get the numbers every time you use the application.

Table 3- System parameters used in the proposed application.

Parameter	Value	Source in code
Secret code space	6 digits, 900,000 codes (100000 to 999999)	generateSecretCode()
Failed-attempt lockout	3 attempts (configurable)	attemptPickup / SystemSettings
Face embedding size	192 values	FaceRecognizer
Face model input	112 x 112 x 3	FaceRecognizer
Face match threshold	0.62 (cosine)	FaceRecognizer.THRESHOLD
Liveness (eye-open) threshold	0.40	FaceRecognizer
Offline sync retries	3 attempts	SyncWorker
Unique identifier format	SDB-YYYY-NNNNNN	UniqueIdGenerator

4.3 Security Analysis

The pickup credential is a six-digit code that is chosen from a range of numbers from 100000 to 999999. This means there are 900,000 codes that can be used. If someone tries to guess the code without knowing it, they will get it right one time in 900,000. This is a small chance, about 1.11×10^{-6} . The slot will lock after a few guesses, which is three by default. So, the chance that someone can guess the code and open the slot before it locks is 3.33×10^{-6} [13]. Table 4 and Fig. 4 show what happens to this chance when more guesses are allowed. The

pickup credential has a face check which is a separate thing. So even if someone figures out the credential code, they still cannot open the slot unless their face matches the owner's face. The face check and the pickup credential code are, like two locks, the pickup credential and the face check [14].

Table 4- Brute-force success probability of the six-digit pickup code before lockout.

Allowed guesses (n)	Success probability
1	1.11×10^{-6}
2	2.22×10^{-6}
3 (default lockout)	3.33×10^{-6}

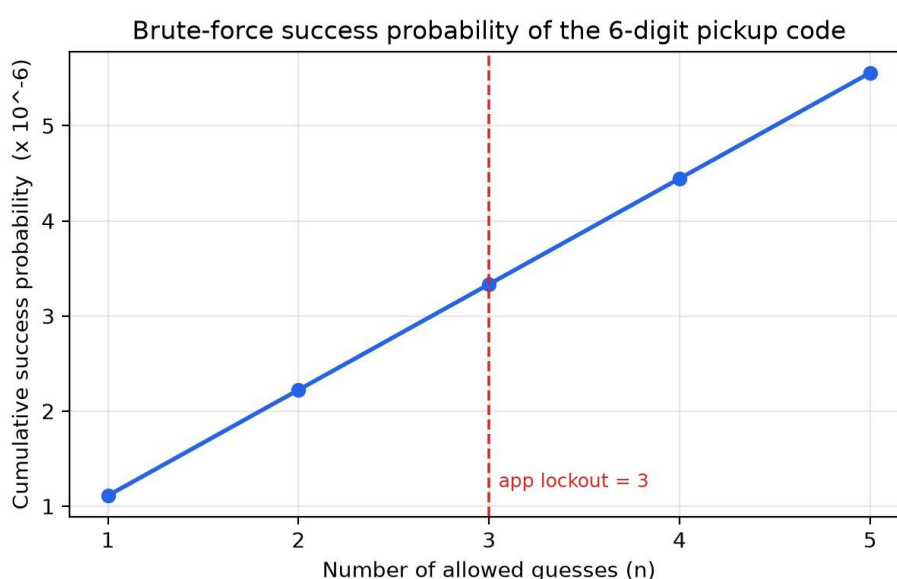


Figure- 4 Cumulative brute-force success probability of the pickup code versus the number of allowed guesses.

4.4 Backend Operation Cost

Because the application does not yet record wall clock timings, the cost of each workflow is described structurally, by the number of Firestore documents writes it performs. A deposit performs three writes: the slot, the notification, and the log. A successful pickup performs two writes, the slot and the log, and so does a pickup with a wrong code. A pickup that reaches the lockout performs three writes: the slot, the log, and the administrator alert. Table 5 and Fig. 5 summarize these counts. The workload of a single service point is small, in order of tens to hundreds of deposits per day, which is well within the documented writing limits of Firestore, so the backend is not expected to be a bottleneck. Measuring the real end to end latency on a device is a direct extension of this work, since it only requires adding timing around the calls that already exist.

Table 5- Backend document writes performed by each workflow.

Workflow	Firestore writes	Documents written
Deposit	3	slot, notification, log
Pickup (success)	2	slot, log
Pickup (wrong code)	2	slot, log
Pickup (lockout)	3	slot, log, admin alert

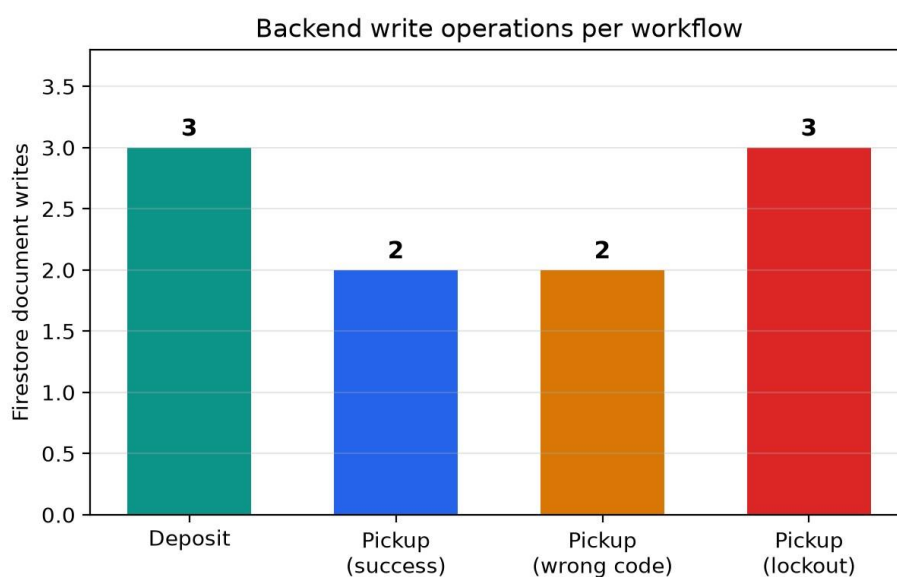


Fig. 5. Number of Firestore document writes per workflow.

4.5 Notification Triggers

The application drives all of its email and pushes notifications from the creation of a notification document. Table 6 lists the notification types defined in the system and the event that triggers each one.

Table 6- Notification types defined in the proposed application and their triggering events.

Type	Recipient	Trigger
CARD_FOUND	Owner	A registered tag is deposited and matched to its owner.
UNKNOWN_TAG_DEPOSITED	Administrators	A tag with no record in tags is deposited; manual review needed.
REMINDER	Owner	A document has been waiting beyond a threshold without pickup.

FINAL_WARNING	Owner	Last notice before the long-stay policy escalation.
ADMIN_ALERT	Administrators	Failed-attempts limit reached, slot moved to DISABLED.
LOST_REPORT_NEW	Administrators	A citizen submits a new lost-document report.
REPORT_UPDATE	Owner	Status of the citizen's lost report changes.
WELCOME	Owner	On account creation; carries the assigned unique identifier.

4.6 Limitations

The present work has three main limitations. First, the application depends entirely on the central Firebase backend, so a long outage of the cloud provider degrades it to the offline mode, where new writes are queued, but already registered documents cannot be matched. Second, the on-device face check depends on lighting and camera quality, which is one reason it is paired with the secret code rather than used on its own. Third, the application does not yet record its own accuracy or response time; reporting measured figures for the face verification accuracy and the end-to-end latency needs a dedicated measurement run, which is left for future work. Two smaller items are the absence of pagination on long lists, where a fixed limit is used, and the fact that Firebase App Check is not yet enabled [14].

5. Conclusions

This paper presented a national android application that will be developed into a complete IoT system for the crowdsourced recovery of lost personal documents. The current implementation consists of the developed Android application and the Firebase cloud backend, while the physical IoT layer, including the NVIDIA Jetson Nano, sensors, and actuators, is considered as future work. In addition to the cloud computing layer, where Firebase from Google is used which is linked to a mobile application for Android devices, which was designed using Kotlin. After an ID is placed at the service point box, it reads the NFC tag and sends a message containing a password for the purpose of document recovery and the location of this service point. After that, the status of the system and the service box will be updated, such as the number of lost and empty slots. One of the features of the application that deserves to be mentioned is that the application supports synchronization operations with the database, whether connected to the Internet or not. A working Android prototype has been validated against eight functional scenarios; the physical box is the subject of ongoing work.

Future work proceeds along three axes. First, the physical prototype on Jetson Nano will be completed and quantitative measurements of latency, throughput, energy consumption, and NFC reliability will be reported. Second, the Jetson Nano's GPU will be

exploited for two edgeAI extensions: a face-capture module that records the depositor for accountability without uploading raw video to the cloud, and a document-type classifier that disambiguates the document category in the UNKNOWN_TAG case. Third, a multi-box deployment across multiple Iraqi governorates is planned, with the goal of validating the per-branch security scoping at scale and producing the first quantitative evidence on the social impact of an automated lost-and-found infrastructure.

Ethical Responsibilities

The authors declare that the manuscript has not been submitted to more than one journal for simultaneous consideration and has not been previously published in part or in full. No data has been fabricated or manipulated to support the conclusions, and no data, text, or theories by others are presented as the authors' own. All sources are properly cited following the IEEE numbered style. The authors who appear on the submission have all contributed sufficiently to the scientific work and share collective responsibility for its results.

This article does not contain any studies involving animals or human subjects performed by any of the authors.

Conflict of Interest

The authors declare that they have no conflicts of interest.

References

- [1] D. Serpanos and M. Wolf, *Internet-of-Things (IoT) Systems*. Cham, Switzerland: Springer, 2018.
- [2] Y. Song, F. R. Yu, L. Zhou, X. Yang, and Z. He, "Applications of the Internet of Things (IoT) in Smart Logistics: A Comprehensive Survey," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4250–4274, Mar. 2021.
- [3] T. Wang, "Internet of Things and Big Data in Smart Logistics," in *Proc. 2022 2nd Int. Conf. Education, Information Management and Service Science (EIMSS)*, Wuhan, China, 2022, pp. 1265–1271.
- [4] R. Want, "Near Field Communication," *IEEE Pervasive Computing*, vol. 10, no. 3, pp. 4–7, Jul.–Sep. 2011.
- [5] V. Coskun, K. Ok, and B. Ozdenizci, *Near Field Communication (NFC): From Theory to Practice*. Hoboken, NJ, USA: Wiley, 2012.
- [6] I. H. Park, J. G. Lee, J. S. Park, S. J. Kim, and J. H. Park, "NFC-LMS: Design and Implementation of Lost Article Management System with NFC," in *Future Information Technology*, Lecture Notes in Electrical Engineering, vol. 164. Dordrecht, The Netherlands: Springer, 2012, pp. 399–406.
- [7] A. A. Phalak, P. N. Jha, N. C. Thoutam, and P. V. Rahane, "An IoT Based Smart Locker Using BLE Technology," *International Journal of Engineering Research & Technology (IJERT)*, vol. 8, no. 5, 2019.
- [8] A. A. Alzhrani, "Design and Implementation of an IoT-Integrated Smart Locker System utilizing Facial Recognition Technology," *Engineering, Technology & Applied Science Research*, vol. 14, no. 4, pp. 16000–16010, Aug. 2024.
- [9] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A Survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

- [10] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of Things: Vision, Applications and Research Challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, Sep. 2012.
- [11] L. Yang, J. Cao, W. Zhu, and S. Tang, "Accurate and Efficient Object Tracking Based on Passive RFID," *IEEE Transactions on Mobile Computing*, vol. 14, no. 11, pp. 2188–2200, Nov. 2015.
- [12] S. Sethuraman, A. Bansal, S. Mardan, M. G. C. Resende, and T. L. Jacobs, "Amazon Locker Capacity Management," *INFORMS Journal on Applied Analytics*, vol. 54, no. 6, pp. 455–470, Nov./Dec. 2024, doi: 10.1287/inte.2023.0005.
- [13] N. Mostakim, R. R. Sarkar, and M. A. Hossain, "Smart Locker: IOT based Intelligent Locker with Password Protection and Face Detection Approach," *International Journal of Wireless and Microwave Technologies (IJWMT)*, vol. 9, no. 3, pp. 1–10, May 2019.
- [14] A. Rahman, L. P. Pratama, Hamzah, B. W. Dianova, A. O. Putri, S. Wilyanti, and S. N. Saud, "Smart Keyless Locker Design Using Face Recognition Technology Based on the Internet of Things Jakarta Global University Classroom," *Journal of Global Engineering Research and Science*, vol. 4, no. 2, pp. 56–66, Dec. 2025.
- [15] Google LLC, "Cloud Firestore Documentation," 2026. [Online]. Available: <https://firebase.google.com/docs/firestore>.
- [16] Google LLC, "Firebase Cloud Messaging Documentation," 2026. [Online]. Available: <https://firebase.google.com/docs/cloud-messaging>.
- [17] JetBrains s.r.o., "Kotlin Programming Language Documentation," 2026. [Online]. Available: <https://kotlinlang.org/docs>.
- [18] Google LLC, "Jetpack Compose Documentation," 2026. [Online]. Available: <https://developer.android.com/jetpack/compose>.
- [19] NXP Semiconductors, "PN532 NFC Controller Datasheet, Rev. 3.6," 2017.
- [20] M. Hartmann and S. Wolthusen, "Authentication in Cloud-Driven IoT-Based Big Data Environments: Survey and Outlook," *Journal of Systems Architecture*, vol. 97, pp. 185–196, 2019.
- [21] Google LLC, "Firebase App Check Documentation," 2026. [Online]. Available: <https://firebase.google.com/docs/app-check>.
- [22] ISO/IEC, "ISO/IEC 14443: Identification cards Contactless integrated circuit cards Proximity cards," International Organization for Standardization, 2018.