Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Campus Lost and Found Portal with Automated Item Matching and Claim Verification: Matching Accuracy and Security Evaluation of a Firebase Web Application at a Nigerian University

Nnaemeka Ugwumba

nnaemeka_ugwumba@uniport.edu.ng

University of Port Harcourt https://orcid.org/0009-0000-2493-9846

Clement Barisua Ndeekor

clement.ndeekor@uniport.edu.ng

University of Port Harcourt, Port Harcourt, Nigeria

Uka Victory

University of Port Harcourt, Port Harcourt, Nigeria

Research Article

Keywords: campus lost and found system, lost and found portal, automated item matching, record linkage, Jaccard similarity, text similarity, sentence embeddings, claim verification, ownership verification, Firebase security rules, Cloud Firestore, access control, web application security, University of Port Harcourt, Nigeria

Posted Date: November 23rd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11803066/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Campus Lost and Found Portal with Automated Item Matching and Claim Verification: Matching Accuracy and Security Evaluation of a Firebase Web Application at a Nigerian University

Clement Barisua Ndeekor
clement.ndeekor@uniport.edu.ng
Department of Computer Science, Faculty of Computing
University of Port Harcourt, Port Harcourt, Nigeria

Nnaemeka Kingsley Ugwumba
nnaemeka_ugwumba@uniport.edu.ng
Department of Computer Science, Faculty of Computing
University of Port Harcourt, Port Harcourt, Nigeria

Uka Victory
ukavictory20@gmail.com
Department of Computer Science, Faculty of Computing
University of Port Harcourt, Port Harcourt, Nigeria

Abstract

Students lose phones, identity cards, calculators, keys and bags every day on a large campus, and at the University of Port Harcourt, Nigeria, reports of lost and found items are scattered across security post logbooks, notice boards and WhatsApp groups. This paper presents the Campus Lost and Found Portal, a web application built on Firebase Authentication, Cloud Firestore and Cloudinary that lets students report lost and found items, browse listings, receive automatic match alerts and file ownership claims that administrators verify against private details recorded by the finder. We then evaluate the two features on which such a portal depends: automated matching and claim verification. We rebuilt the portal's matching algorithm exactly, a rule score out of 100 based on category, word overlap and date, and tested it on 375 lost and 375 found reports per test set with known true matches, written in the varied ways owners and finders describe the same item and set at real University of Port Harcourt places. At the deployed threshold of 50 the algorithm found 97 percent of true matches but sent 27 alerts per lost report, of which only 2.4 percent were correct, and false alerts grew in proportion to the number of open reports, reaching 150 per lost report with 2,000 open found reports. A gradient boosted model that combines field level similarities ranked the true item first for 87 percent of lost reports against 43 percent for the deployed rule, sent 0.6 alerts per report and was right 77 percent of the time. Sentence embeddings from MiniLM, BM25 and TF IDF alone performed far below the combined model. A security review of the deployed Firestore rules found that found item documents, including the private marks and exact location used to verify ownership, are readable by anyone. A fake lost report copied from a public listing passed the match threshold for every found item, and an impostor who read the private fields matched them perfectly in every case, while an impostor limited to the public listing was separated from true owners with a ROC AUC of 0.81. We propose and test a revision in which private fields live in owner and administrator only collections, match suggestions and notifications are checked by the database rules, and a public field matching rule alerts only the best candidate; this rule sent 0.7 alerts per lost report with 39 percent precision and placed the true item in the top five for 81 percent of reports.

Keywords: campus lost and found system, lost and found portal, automated item matching, record linkage, Jaccard similarity, text similarity, sentence embeddings, claim verification, ownership verification, Firebase security rules, Cloud Firestore, access control, web application security, University of Port Harcourt, Nigeria

1 Introduction

A university campus is a place where personal belongings go missing every day. Students move between lecture halls, laboratories, libraries, hostels and sports grounds, and phones, identity cards, calculators, textbooks, keys, bags and clothing are left behind along the way. The loss is rarely only the item. A lost laptop may hold a semester of coursework, a missing identity card can block entry to an examination hall, and a lost hostel key may leave a student locked out until a replacement is found. For students on tight budgets, replacing a phone or a scientific calculator is a real hardship.

At the University of Port Harcourt (UNIPORT), as at many Nigerian universities, lost property is handled through channels that do not talk to each other. A finder hands an item to the nearest security post, a departmental office or a lecturer, where it is written into a paper register. A student who has lost something visits offices, scans notice boards and posts in WhatsApp, Telegram and Facebook groups. These channels are fast, but a message posted at nine in the morning can be buried by noon, and a found item recorded at one faculty is invisible to a student searching at another. When an item is finally returned, ownership is usually established by a verbal description that nobody can check.

A web portal can bring these reports into one place. The Campus Lost and Found Portal described in this paper was built for UNIPORT on Google Firebase and Cloudinary. Students register, report lost and found items with photographs, browse active listings, receive a notification when the system finds a possible match, and file a claim that an administrator reviews side by side with the lost

report and the finder's private notes before an item is released.

Two features carry most of the value of such a portal and most of its risk. The first is automated matching: if it misses true matches, owners are never told; if it flags too many false matches, owners learn to ignore its alerts. The second is claim verification: if an impostor can describe an item as well as its owner, the portal becomes a convenient way to collect other people's property. Functional testing, which confirms that a match is created when two similar reports are entered, does not measure either property. This paper measures both.

We ask four questions.

1. How should a low cost campus lost and found portal be designed for students and administrators, and how was the UNIPORT portal built?

2. How accurate is the portal's rule based matching algorithm, how does accuracy change as reports become less precise and as the number of open reports grows, and which parts of the rule matter?

3. How do established text matching methods and a learned combination of field similarities compare with the deployed rule?

4. How well does the claim process resist impostors, and what does the deployed database configuration expose?

Our contributions are:

1. the design and implementation of a Firebase based campus lost and found portal with structured reporting, public and private report fields, automated match alerts, claim verification and role based administration;

2. a test collection of lost and found reports with known true matches, set at real UNIPORT places and written with the variations seen between owners and finders, together with an exact reimplementation of the portal's matching rule;

3. a comparison of seven matching methods, including the deployed rule, TF IDF, BM25, MiniLM sentence embeddings and a learned combiner, at three levels of report noise, with threshold, component and scale analyses;

4. a security review of the deployed Firestore rules and an experiment on claim fraud, showing that the private verification fields are publicly readable and that the match threshold is no barrier to impostors;

5. specific recommendations for matching, alerting, verification and access rules.

Section 2 reviews related work. Section 3 describes the existing process at UNIPORT. Section 4 presents the system design and Section 5 the security analysis. Section 6 describes the evaluation method and Section 7 the results. Section 8 discusses implications and limitations, and Section 9 concludes.

2 Literature Review

2.1 Lost and found systems for campuses

Several institutions have moved lost and found work from registers and chat groups to web systems. David and colleagues compared lost and found management systems in academic institutions and found that most combined a reporting form, a searchable list and an administrator view, with large differences in how ownership was checked [4]. Pede and colleagues built a lost and found web application for a university campus in which users post lost or found items, search with filters, receive notifications of matching items and rely on an administrator panel to verify posts [18]. At Universiti Tun Hussein Onn Malaysia, Shahzan and Arbaiy replaced the informal Telegram and WhatsApp channels used by students with a central web system for reporting, matching and administering found items [25], and Murugiah and Mat Nawi built FindItHere, a lost and found web application with real time notifications for matched items [15]. These systems show that the problem is common and that students accept web portals for it. None of them reports how accurate its matching is or how well its claim process resists false claims.

2.2 Matching records that describe the same object

Deciding whether a lost report and a found report describe the same item is an instance of record linkage, the task of finding records that refer to the same entity when no shared identifier exists. Fellegi and Sunter gave the classical probabilistic theory, in which agreement on each field contributes evidence for or against a match [8]. Elmagarmid, Ipeirotis and Verykios surveyed duplicate record detection and the field comparison methods it uses [5], and Christen described the full process of data matching, from cleaning and normalising fields to comparison, classification and evaluation [3].

At the field level, Jaccard's coefficient measures the overlap of two sets as the size of their intersection divided by the size of their union [12], and Broder defined containment, the share of one set found in the other, for comparing documents of different length [2]. The UNIPORT portal uses the larger of the two. Information retrieval offers weighted alternatives: TF IDF weighting with cosine similarity [13] and the BM25 ranking function [22], both of which give rare words more influence than common ones. Approximate string matching tolerates misspellings [16]. Sentence embeddings, such as those from Sentence BERT [21] and the compact MiniLM model [27], place texts with similar meaning close together even when they share no words.

2.3 Verifying ownership with knowledge

A lost and found claim asks the claimant to prove ownership by describing something only the owner should know, such as a scratch, a sticker or the contents of a bag. This is a form of knowledge based authentication. Schechter, Brush and Egelman found that answers to personal secret questions were often guessable by

acquaintances and forgotten by their owners [24], and Bonneau and colleagues, studying hundreds of millions of secret question answers at Google, found that common answers could be guessed with few attempts while many users could not recall their own [1]. Both weaknesses apply to item descriptions: common items have common marks, and owners do not always remember the details a finder notices.

2.4 Access control in cloud backends

Role based access control assigns permissions to roles rather than individuals [23]. Backend as a Service platforms such as Firebase let a browser application read and write a cloud database directly [14], so access control must be enforced by security rules on the server rather than by the user interface. Firebase documentation states that rules are not filters: a rule grants or denies access to whole documents, so any field stored in a readable document is readable by any client that can query it [10]. Broken access control is ranked first among web application risks by OWASP [17].

2.5 Research gap

Table 1 places this study among the closest prior work. Campus lost and found systems have been built and described, but their matching has not been measured against known true matches, their claim processes have not been tested against impostors, and their database rules have not been examined for the exposure of private verification data. This paper does all three for a deployed portal.

Table 1. Prior work closest to this study

Work	Setting	What it provides	What it does not address
David et al. [4]	academic institutions	comparison of lost and found management systems	measured matching accuracy
Pede et al. [18]	university campus	web reporting, search, match notifications, admin verification	accuracy of match notifications, claim fraud
Shahzan and Arbaiy [25]	UTHM, Malaysia	central system replacing chat groups	matching evaluation, access rules
Murugiah and Mat Nawi [15]	web application	item matching with real time notifications	comparison of matching methods
Fellegi and Sunter [8], Christen [3]	record linkage	theory and practice of matching records	lost and found reports, ownership checks
Schechter et al. [24], Bonneau et al. [1]	secret questions	guessability and recall of knowledge answers	item descriptions as proof of ownership

3 The Existing Process at UNIPORT

Before the portal, lost property at UNIPORT moved through four disconnected channels: security posts with handwritten logbooks, departmental offices, physical notice boards, and social media groups. Each has value, but together they produce four problems. Information is scattered, so a found item held at the Faculty of Science security post is invisible to a student who lost it near the Faculty of Engineering. Visibility is lost quickly, because chat messages are pushed out of view within hours and cannot be searched reliably. Ownership is not verified, because the finder's observations are rarely recorded and the claimant's description cannot be checked against anything. And the administrative burden is high, because staff must file and cross check paper registers by hand while unclaimed items accumulate in storage. Table 2 compares this process with the portal.

Table 2. Existing process and the portal

Aspect	Existing process	Portal
Reporting	verbal, notice boards, chat posts	structured forms with photo
Storage	paper logbooks, chat history	Cloud Firestore
Search	visit offices, scroll chats	keyword and category search
Matching	chance	automatic score and alert
Private details	not recorded	finder records hidden fields
Verification	verbal description	admin compares proof with private fields
Records	none	full history of reports and decisions

4 System Design

4.1 Requirements

The portal was designed to meet seven objectives: secure registration and login; forms for reporting lost and found items with optional images; a central cloud database for users, reports, claims, match suggestions and notifications; automated matching across category, item name, description, location and date; a claim workflow in which administrators compare proof with the linked reports; role based access enforced in the interface and in database rules; and a responsive interface for phones and computers. Each user may hold at most five open reports in each collection, which limits spam.

4.2 Architecture

Figure 1 shows the three tier architecture. The client tier is a set of HTML5, CSS3 and JavaScript pages, with a shared module that handles routing, validation and the matching algorithm. The communication tier uses the Firebase JavaScript SDK over HTTPS and the Cloudinary REST interface. The backend tier consists of Firebase Authentication for identity, Cloud Firestore for data, and Cloudinary for image storage and delivery. Cloudinary was chosen because Firebase Storage required a paid plan; uploaded image addresses are rewritten to request automatic format and quality, so phones on slow networks receive smaller files. The site is deployed on Cloudflare Workers and rebuilt on every push to its GitHub repository.

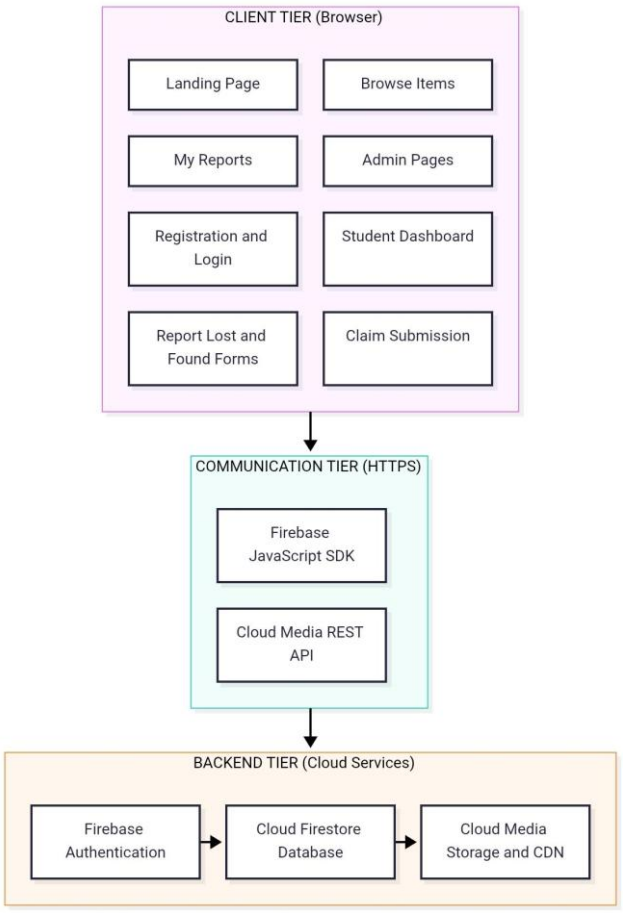

Figure 1. Three tier architecture of the portal.

4.3 Data model

Figure 2 shows the entity relationship diagram. Six Firestore collections hold the data, linked by document identifiers rather than nesting (Table 3). A document model suits the different shapes of the reports: a found report carries private custody details and an administrator note that a lost report does not [6]. The split between public and private fields on found reports is central to the design. Item name, category, general description, date and photograph are shown on public listings. The exact location found, the hand in location, the unique identifying marks and the administrator note are meant to be seen only by administrators, so that a claimant must supply them from their own knowledge.

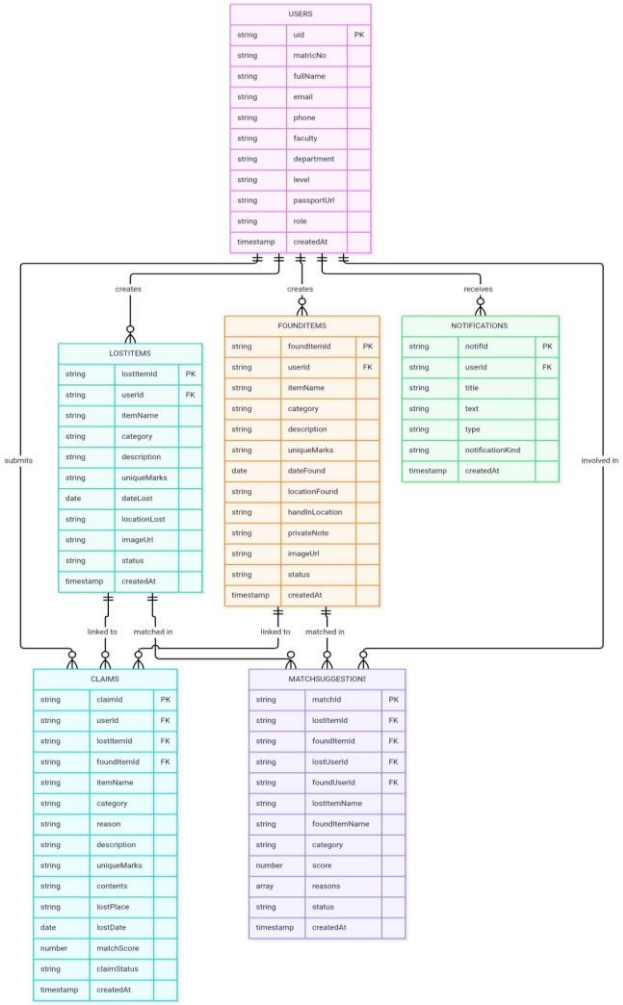

Figure 2. Entity relationship diagram of the six Firestore collections.

Table 3. Firestore collections

Collection	Main fields
users	matriculation number, name, email, phone, faculty, department, level, photo, role
lostItems	item name, category, description, unique marks, date lost, location lost, image, status
foundItems	public: item name, category, description, date found, image; private: exact location, hand in location, unique marks, admin note
claims	claimant, lost and found report links, proof fields, match score, status
matchSuggestions	lost and found report links, score, reasons, status
notifications	recipient, title, text, kind (possible match, claim decision, item returned)

4.4 Actors and use cases

Figure 3 shows the use cases. Students register, log in, report lost and found items, browse and search, receive notifications, submit ownership claims, manage their reports and settings, and reset passwords. Administrators monitor the dashboard, manage reports, review pending claims, approve or reject them, and inspect match suggestions.

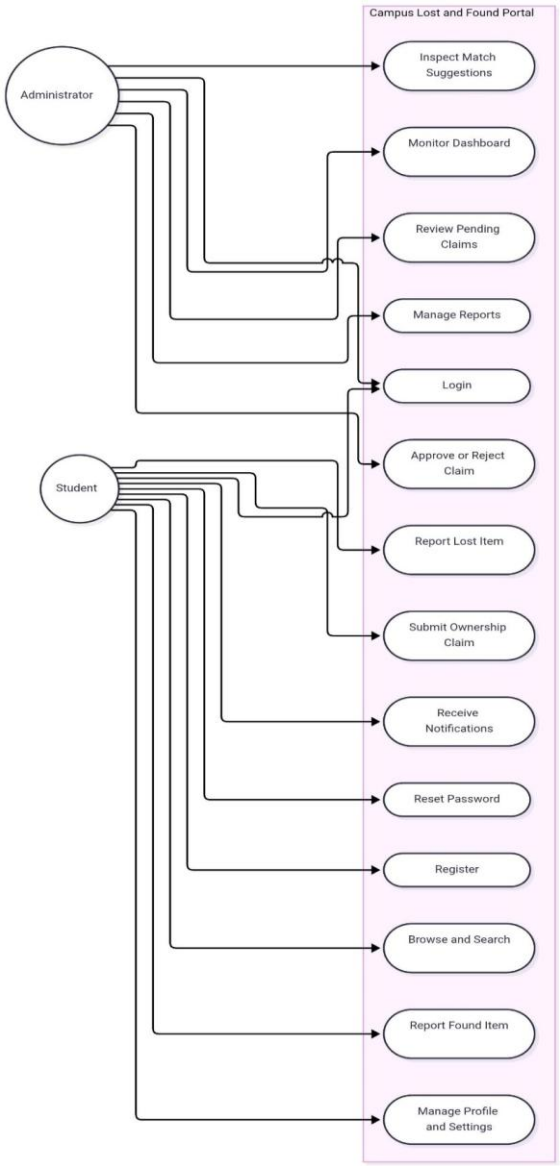

Figure 3. Use case diagram of the portal.

4.5 The matching algorithm

Each time a report is saved, the browser compares it with every open report in the opposite collection and computes a score out of 100 (Figure 4, Table 4). Text is lowercased, punctuation is removed, words of two letters or fewer and a short list of stop words, including lost, found and item, are dropped, and two texts are compared by the larger of their Jaccard similarity and their containment. If the score reaches 50, the portal stores a match suggestion with its reasons and notifies the owner of the lost report. The finder is not notified. The same rule fills the list of found items a student may claim against.

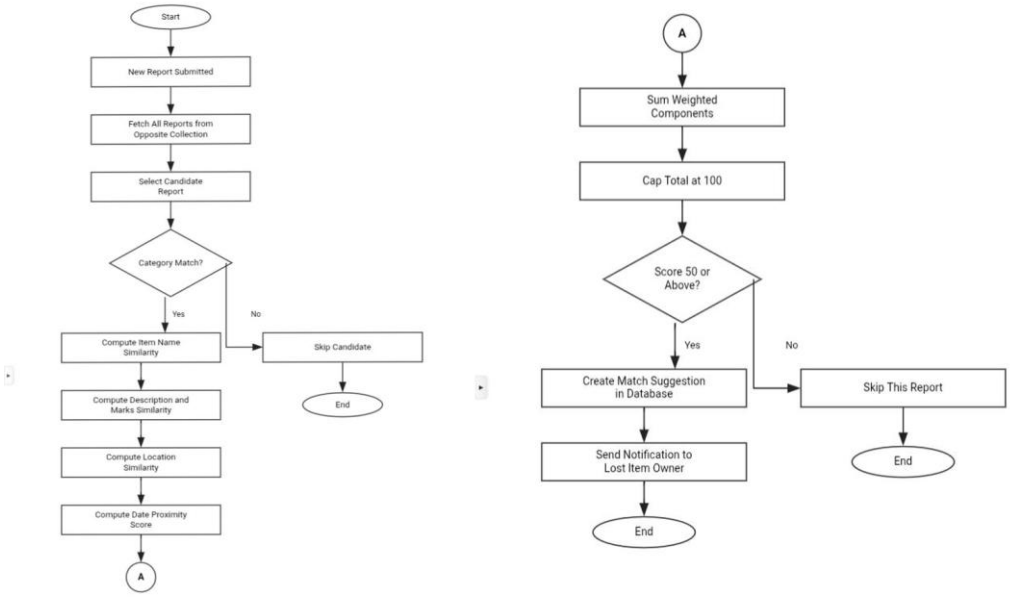

Figure 4. Flowchart of the automated matching algorithm (left: comparison steps; right: scoring and alert).

Table 4. Points awarded by the deployed matching rule

Component	Condition	Points
Category	same category	35
	differ, one is Others	10
	differ otherwise	stop, score 0
Item name	similarity at least 0.5	35
	0.25 to 0.5	20
Description and marks	at least 0.3	20
	0.12 to 0.3	10
Location	at least 0.3	10
Date	within 2 days	10
	within 7 days	7
	within 14 days	4
Alert	total at least 50	notify

The rule has an important property that the results in Section 7 confirm. Because containment equals 1 whenever every word of the shorter text appears in the longer one, an item name of one meaningful word, such as phone, matches any other name containing that word. Two reports in the same category with a shared name word therefore reach 70 points before description, location or date are considered.

4.6 Claim verification

A student who believes a found item is theirs opens the claim page, chooses one of their open lost reports, and then chooses from the found reports that score at least 50 against it. The claim form asks why the student believes the item is theirs, a full description, unique marks, contents and accessories, and where and when it was lost. Duplicate pending claims on the same item are blocked. Administrators see each pending claim as three columns: the claimant's proof, the original lost report, and the found report with its private fields. Approval marks the claim approved, the lost report resolved, the found report claimed and the match suggestion archived, and notifies both the claimant and the finder. Rejection leaves both reports open for other claims.

4.7 User interface

Figures 5 to 9 show the implemented interface, which uses a dark theme and works on phones and computers [20]. Registration collects the matriculation number, name, phone, email, faculty, department, level, an optional passport photograph and a password of at least six characters, and the login page offers password reset (Figure 5). The student dashboard links to every task, and the browse page lists active lost and found items with keyword search and a category filter (Figure 6). The lost report form asks for the item name, category, a detailed description, unique marks, date, location and a photograph; the found report form separates the general public description from confidential unique marks, the exact location found, the custody location and a private note (Figure 7). The claim page offers only algorithm matched found items, and the My Reports page lets students edit and close their reports and track claims (Figure 8). The administrator dashboard counts lost and found reports, pending claims and open match suggestions; the claims verification page shows the three column comparison; and the possible matches page lists suggestions scoring 50 or more (Figure 9).

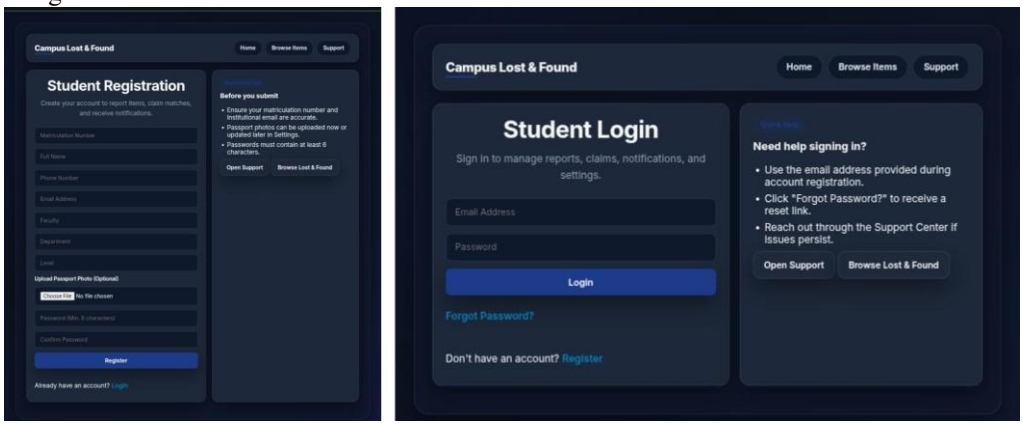

Figure 5. Student registration page (left) and login page (right).

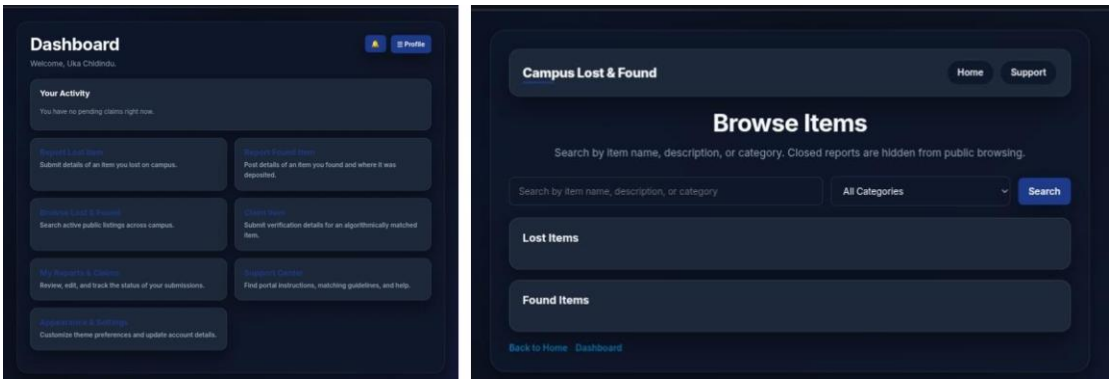

Figure 6. Student dashboard (left) and public browse page with search and category filter (right).

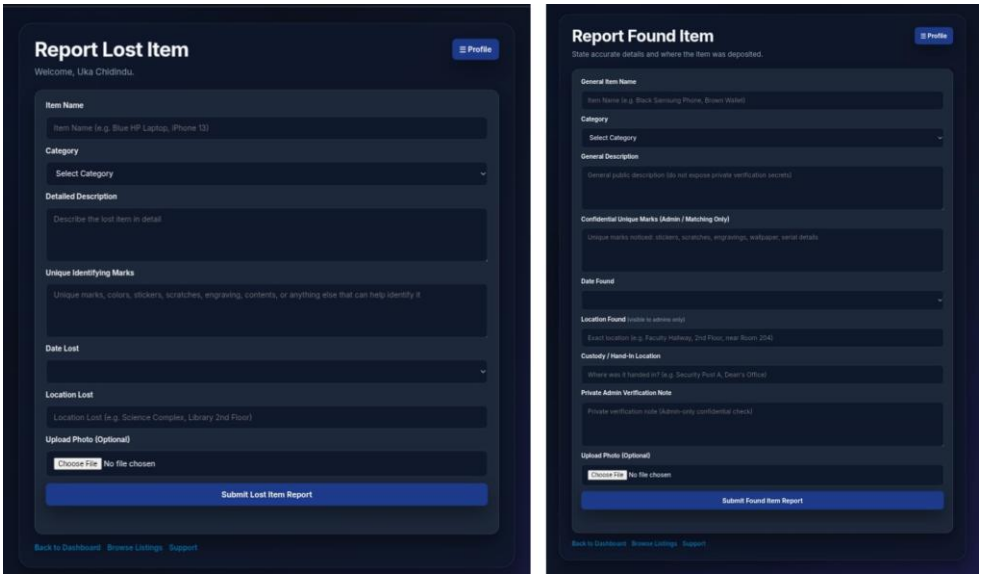

Figure 7. Report lost item form (left) and report found item form with confidential fields (right).

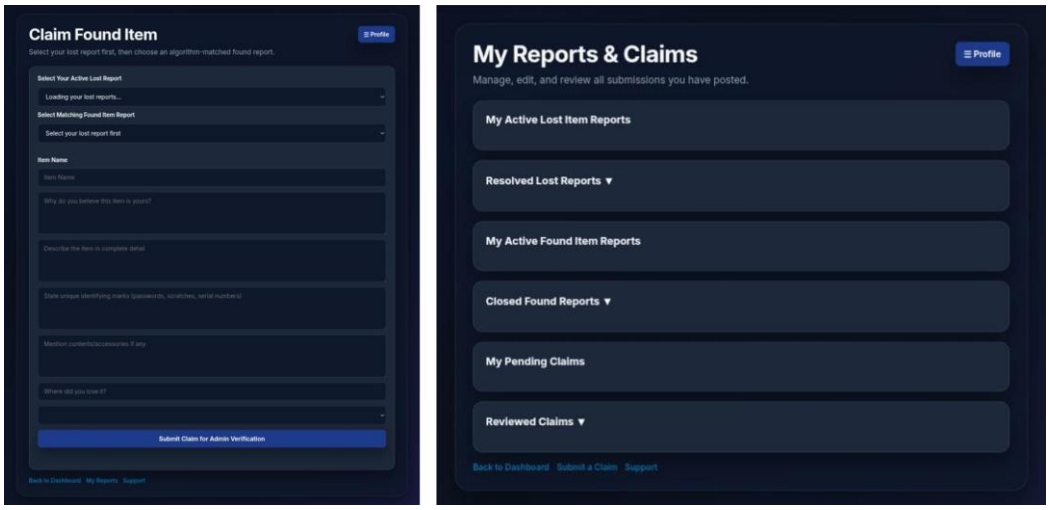

Figure 8. Claim submission page limited to matched found items (left) and the My Reports and Claims page (right).

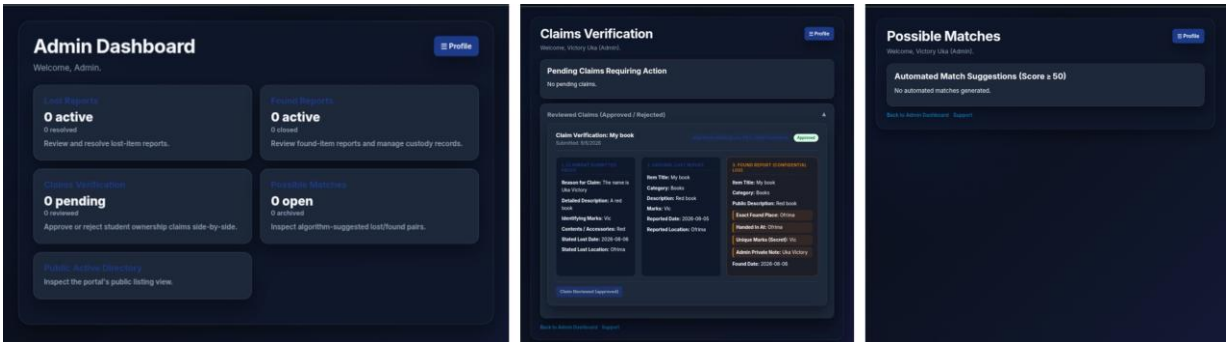

Figure 9. Administrator dashboard (left), claims verification with proof, lost report and confidential found report side by side (centre), and the possible matches page (right). Test accounts are shown.

4.8 Functional testing

The developers followed an incremental process [26] and tested the portal with unit, integration, system and acceptance tests. Table 5 summarises the reported test cases, all of which passed. These tests confirm that each function behaves as designed; they do not measure how often the match alerts are right or whether the claim process can be fooled, which Sections 5 to 7 address.

Table 5. Functional tests reported by the developers

Area	Tests (all passed)
Accounts	registration, mismatched passwords blocked, login, role routing
Reporting	valid lost and found reports saved, sixth open report blocked
Matching	similar reports create a suggestion and notify the owner; unrelated reports in the same category create none
Search	keyword in description found; category filter
Claims	dropdown shows only items scoring 50 or more; approval updates four records and notifies both users; rejection keeps reports open
Privacy	public browse hides location, marks and admin note
Closure	owner marks a report returned

5 Security Analysis

5.1 Deployed database rules

Figure 10 shows the Firestore security rules deployed with the portal. They define helper functions for signed in users, administrators and owners, validate categories, and protect users, claims and notifications so that only the owner or an administrator can read them. Users cannot change their own role. Two rules, however, undermine the privacy design. The lostItems and foundItems collections both allow read access to anyone, including visitors who are not signed in. And the matchSuggestions collection allows any signed in user to create and update suggestions, while the notifications collection allows any signed in user to create a notification for any other user.

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {

    /* ---------- HELPER FUNCTIONS ---------- */
    function isSignedIn() {
      return request.auth != null;
    }

    function isAdmin() {
      return isSignedIn()
        && exists(/databases/$(database)/documents/users/$(request.auth.uid))
        &&
get(/databases/$(database)/documents/users/$(request.auth.uid)).data.role ==
'admin';
    }

    function isOwner(userId) {
      return isSignedIn() && request.auth.uid == userId;
    }

    function isValidCategory(cat) {
      return cat in ['Electronics', 'Books', 'ID Card', 'Keys', 'Bags',
'Clothes', 'Others'];
    }

    /* ---------- USERS ---------- */
    match /users/{userId} {
      // Anyone signed in can read their own profile; admins can read any.
      allow read: if isSignedIn() && (isOwner(userId) || isAdmin());

      // Users can only create their own profile, forced role = "student".
      allow create: if isSignedIn()
        && isOwner(userId)
        && request.resource.data.role == 'student'
        && request.resource.data.matricNo is string
        && request.resource.data.fullName is string
        && request.resource.data.email is string;

      // Users can update their own profile, but NOT change their role.
      // Admins can update anyone (e.g. promote a user).
      allow update: if isSignedIn()
        && (
          isAdmin()
          || (
            isOwner(userId)
            && request.resource.data.role == resource.data.role
          )
        );

      // Only admins can delete users.
      allow delete: if isAdmin();
    }

    /* ---------- LOST ITEMS ---------- */
    match /lostItems/{itemId} {
      // Public read – guests can browse lost listings.
      allow read: if true;

      // Only signed-in users can create; userId must match their own UID.
      allow create: if isSignedIn()
        && request.resource.data.userId == request.auth.uid
        && request.resource.data.itemName is string
        && request.resource.data.itemName.size() > 0
        && isValidCategory(request.resource.data.category)
        && request.resource.data.status == 'open';

      // Owners can update their own reports (status transitions to resolved).
      // Admins can update any report.
      allow update: if isSignedIn()
        && (
          isAdmin()
          || (
            resource.data.userId == request.auth.uid
            && request.resource.data.userId == resource.data.userId
          )
        );

      allow delete: if isSignedIn()
        && (isAdmin() || resource.data.userId == request.auth.uid);
    }

    /* ---------- FOUND ITEMS ---------- */
    match /foundItems/{itemId} {
      allow read: if true;

      allow create: if isSignedIn()
        && request.resource.data.userId == request.auth.uid
        && request.resource.data.itemName is string
        && request.resource.data.itemName.size() > 0
        && isValidCategory(request.resource.data.category)
        && request.resource.data.status == 'open';

      allow update: if isSignedIn()
        && (
          isAdmin()
          || (
            resource.data.userId == request.auth.uid
            && request.resource.data.userId == resource.data.userId
          )
        );

      allow delete: if isSignedIn()
        && (isAdmin() || resource.data.userId == request.auth.uid);
    }

    /* ---------- CLAIMS ---------- */
    match /claims/{claimId} {
      // Only the claimant and admins can read a claim.
      allow read: if isSignedIn()
        && (isAdmin() || resource.data.userId == request.auth.uid);

      // Only the claimant can create a claim, and it must start as "pending".
      allow create: if isSignedIn()
        && request.resource.data.userId == request.auth.uid
        && request.resource.data.claimStatus == 'pending';

      // Admins approve/reject. Claimants can edit their own pending claim.
      allow update: if isSignedIn()
        && (
          isAdmin()
          || (
            resource.data.userId == request.auth.uid
            && resource.data.claimStatus == 'pending'
          )
        );

      allow delete: if isSignedIn()
        && (isAdmin() || resource.data.userId == request.auth.uid);
    }

    /* ---------- MATCH SUGGESTIONS ---------- */
    // Students create these when they submit a report. Admins archive them.
    match /matchSuggestions/{matchId} {
      allow read: if isSignedIn();
      allow create: if isSignedIn();
      allow update: if isSignedIn();
      allow delete: if isAdmin();
    }

    /* ---------- NOTIFICATIONS ---------- */
    // Users create notifications for other users (match alerts, claim
decisions).
    // Users can only read/delete their own notifications.
    match /notifications/{notifId} {
      allow read: if isSignedIn()
        && (isAdmin() || resource.data.userId == request.auth.uid);
```

Figure 10. Firestore security rules deployed with the portal.

5.2 Findings

Table 6 lists the findings. The most serious follows from the fact that Firestore rules are not filters [10]. The public browse page hides private fields only in the interface; the found item documents themselves, with exact location, unique marks and the administrator's private note, are returned in full to any client that queries the collection, and the Firebase configuration needed to do so is, by design, in the page source. The matching algorithm makes this unavoidable in the current design, because it runs in the browser and compares lost reports with the private marks and exact location of every found report. The developers noted the first limitation in their report. Section 7.7 measures its consequence for claims.

Table 6. Security findings and recommended changes

Finding	Effect	Severity	Recommended change
foundItems readable by anyone	private marks, exact location and admin notes can be read by any visitor	high	move private fields to a separate collection readable only by administrators
Matching runs in the browser on private fields	every client must download private fields to compute scores	high	run matching in a Cloud Function with administrator privileges; return only scores
Any user can create notifications for others	fake match or claim decision messages; phishing	medium	create notifications only from server functions
Any user can create or update match suggestions	false suggestions in the administrator queue	medium	server side creation; read only for students
Unsigned Cloudinary upload preset	anyone can upload images to the account until quota	low	signed uploads through a server function
Match threshold gates the claim dropdown	impostors can reach any item with a copied report (Section 7.7)	medium	do not treat the match score as evidence of ownership
lostItems readable by anyone	owners' descriptions and marks are public	low to medium	show summaries publicly; keep marks private

The personal data handled by the portal, including names, matriculation numbers, phone numbers and photographs, falls under the Nigeria Data Protection Act [7].

5.3 Proposed revision

We designed a revision that removes these weaknesses without a paid Firebase plan or a separate server. Table 7 lists the changes. Private verification fields move into two new collections, foundItemsPrivate and lostItemsPrivate, keyed by the same document identifier as the public report and readable only by the report's owner and administrators; the rules for foundItems and lostItems reject any write that contains a private field, and a one time migration moves the fields of existing reports. Match suggestions can be created only by the owner of the lost or found report involved and only for reports that exist and belong to the stated users, and students may read only their own suggestions. A user can create a notification only for themselves, or as a possible match alert that names an existing match suggestion in which the recipient owns the lost report. Because the browser can no longer read private fields, matching is rewritten to use public fields only; its design and measured accuracy are reported in Section 7.8. The Cloudinary upload preset is limited by format, size and folder.

Table 7. Proposed changes to the portal

Area	Change
Private fields	moved to foundItemsPrivate and lostItemsPrivate; owner and admin read only; public writes containing them rejected
Existing data	migration script moves fields and deletes them from public documents
Match suggestions	create only by a participant, for existing reports; read only by participants and admins
Notifications	self, admin, or possible match alert tied to a real suggestion
Matching	runs on public fields only; alerts the best candidate; claim page shows top five
Claim review	admin page reads private documents; reminder to ask for unlisted details
Image uploads	preset limited by format, size and folder

6 Evaluation Method

6.1 Why a test collection with known matches

UNIPORT's lost property records are on paper, so no historical set of lost and found reports with confirmed outcomes exists. Public open data on lost property, such as government lists of found items, contains found items only, without the lost reports they were matched to. To measure matching, we therefore built test collections in which the true match of every report is known, and in which owners and finders describe items in the different ways observed in practice.

6.2 Building the test collections

Each item is drawn from a catalogue of 17 item types in the portal's seven categories, weighted by how often they are lost: phones, laptops, power banks, earbuds, calculators, textbooks, lecture notes, identity cards, keys, backpacks, handbags, lab coats, hoodies, wallets, glasses, flash drives and umbrellas. Each type has owner and finder nouns, brands or titles, colours, distinguishing marks and, where relevant, contents. Places are 33 real UNIPORT locations from OpenStreetMap [11], such as the Donald Ekong Library, the Senate Building, Ofrima, the Lulu Briggs Health Centre and the King Jaja and Dan Etete hostels, each with the aliases students use. Some aliases, such as UL for the library, are written by reporters but deliberately left out of the normaliser's list, so the normaliser is not tested only on words it knows.

For each item, the owner's lost report and the finder's found report are written separately. Owners tend to name the brand and model, list marks in their own words and give the place they believe they lost the item, which is sometimes wrong or vague. Finders use generic nouns, may misjudge colour (navy for blue, ash for grey), note only the marks they can see, reword them, may choose a different category, and record the exact place and a date some days later. Table 8 lists the probabilities of these variations at three noise levels, and Table 9 shows example pairs from the medium level, including deliberate misspellings.

Table 8. Report noise settings (probabilities)

Variation	Low	Medium	High
Misspelt word	0.02	0.05	0.10
Colour misjudged by finder	0.05	0.12	0.20
Finder picks other category	0.05	0.12	0.25
Owner names wrong place	0.15	0.30	0.45
Owner gives vague place	0.05	0.10	0.20
Unlisted alias or synonym	0.10	0.25	0.40
Finder records brand	0.70	0.50	0.35
Finder notes each mark	0.75	0.60	0.45
Empty description	0.05	0.12	0.25

Table 9. Example lost and found report pairs describing the same item (medium noise, misspellings as generated)

Side	Item name	Description	Marks	Location	Category
Lost	rimless glasses	it is brown, rimless	tape on the arm; in a brown case	health centre	Others
Found	brown glasses	brown glasses found	tape on the arm; in a bown case	lulu briggs	Others
Lost	school card of Precious Ibe	it is blue, UNIPORT student ID	lanyard attached	pharm	ID Card
Found	blue school id	blue school id found	lanyard attached	pharmacy faculty	Others
Lost	id card of Boma West	it is white, UNIPORT student ID	inside a transparent holder	ofrima	ID Card
Found	id card Boma West	whte identity card found	lanyard attached	ofrima building	ID Card
Lost	hostel room key bunch of keys	it is black, hostel room key	red tag with room number; bottle opener keyholder	shopping complex	Keys
Found	black key bunch	black key bunch found	(none noted)	Uniport Mall	Keys
Lost	school card of Precious Briggs	it is blue, UNIPORT student ID	photo slightly faded; inside a blue card holder	around Choba campus	ID Card
Found	id card Precious B.	blue id card found, UNIPORT	photo slightly faded; inside a blue card holder	AK hostel	ID Card

Each test collection holds 250 items reported by both owner and finder, 125 lost only and 125 found only, giving 375 lost and 375 found reports and 140,625 possible pairs, of which 250 are true matches. Table 10 shows the category mix of one medium noise collection. Found reports filed under Others are twice as common as lost ones because finders often choose Others for items such as calculators, power banks and identity cards. We built three test collections at each noise level with different random seeds, plus separate training and validation collections at medium noise.

Table 10. Reports by category in one medium noise test collection

Category	Lost	Found
Electronics	138	125
Books	43	40
ID Card	51	35
Keys	24	17
Bags	38	43
Clothes	37	26
Others	44	89
Total	375	375

6.3 Matching methods

We compared seven methods. **Portal** is the deployed rule, reimplemented exactly from the portal's code, including the stop word list, the word length filter, the maximum of Jaccard and containment, and the threshold of 50. **Portal with normaliser** keeps the same points but first maps synonyms to one word, colours to base colours and known place aliases to one place, and corrects words one edit away from the vocabulary. **TF IDF words** and **TF IDF characters** compute the cosine similarity of TF IDF vectors over words and pairs of words, or over character sequences of three to five letters, of all report fields together [13]. **BM25** ranks found reports with the Okapi BM25 function [22]. **MiniLM** uses the cosine similarity of sentence embeddings from the all MiniLM L6 v2 model [21, 27]. The **learned combiner** is a gradient boosted tree model [9, 19] over 14 field level features: same category, either category Others, same item type, raw and normalised similarities of name, description and location, date difference, found before lost, same colour, character TF IDF similarity and MiniLM similarity. It was trained on all 250 true pairs and 10,000 sampled non matching pairs from the training collection, most of them from the same category. This follows the record linkage approach of combining field comparisons into one decision [3, 8].

6.4 Measures and thresholds

We measure matching in two ways that reflect how the portal is used. For alerts, a pair is flagged when its score reaches a threshold; we report precision (share of alerts that are true), recall (share of true matches alerted) and alerts per lost report. The portal uses its deployed threshold of 50. For each other method we chose, on the validation collection, the threshold with the best F1 score and, separately, the lowest threshold reaching 80 percent precision (Table 11). For ranking, the found reports are sorted by score for each lost report, as in a list shown to the owner, and we report the share of true matches ranked first (R1) and within the top five (R5), the mean reciprocal rank (MRR) and average precision (AP) over all pairs. Ties, frequent with the portal's coarse points, are broken at random.

Table 11. Thresholds chosen on the validation collection

Method	Best F1	80% precision
Portal	50 (deployed)	100
Portal with normaliser	95	none reached
TF IDF words	0.410	0.585
TF IDF characters	0.596	0.716
BM25	22.14	31.83
MiniLM	0.774	0.847
Learned combiner	0.837	0.880

6.5 Revised matching rule

The revised rule runs in the browser on public fields only. It awards 25 points for the same category and 10 when one report is filed under Others, without stopping the comparison when categories differ; 20 points when the Jaccard similarity of item names is at least 0.5 and 10 when it is at least 0.25, with no containment; 8 points for each shared distinctive word, meaning a word of four or more letters or containing a digit that is not a colour or a generic noun such as phone or card, up to 32; and 20, 12 or 5 points when the item was found within 2, 7 or 14 days of the loss, or 8 when it was found up to 3 days before the loss was reported. The owner's own unique marks are used when the owner's browser runs the comparison. Instead of a fixed threshold for all pairs, the owner is alerted only about the single best candidate, and only if it scores at least 50, and the claim page shows the five best candidates. These settings were chosen on the validation collection.

6.6 Claim experiment

For each of the 250 true items in one medium noise collection, we compared three kinds of claim proof with

the found report's private fields (unique marks and exact location), using the portal's text similarity and the normalised version. The genuine owner's proof contains the marks, description and location from their own lost report. The first impostor sees only the public listing (item name, category, general description) and adds two marks commonly found on that type of item and a busy campus place. The second impostor has read the private fields directly from the database, as the rules allow, and copies them. We also tested whether a fake lost report copied from each public listing would pass the match threshold of 50 and so place the item in the impostor's claim list. The whole experiment runs in one Kaggle notebook cell in about two minutes.

7 Results

7.1 The deployed rule finds almost everything and alerts on almost everything

Table 12 gives the main results at medium noise. At its threshold of 50, the portal's rule alerted 96.8 percent of true matches, but it sent 27.3 alerts for every lost report, and only 2.4 percent of those alerts were true. A student who lost a phone would be told of about two dozen phones, most of them someone else's. As a ranking, the rule placed the true item first for 43 percent of lost reports and within the top five for 71 percent.

Table 12. Matching results at medium noise, mean of three test collections. Precision, recall and alerts per lost report use the deployed threshold for the portal and the validation best F1 threshold for the other methods.

Method	Precision	Recall	Alerts per lost	R1	R5	MRR	AP
Portal (deployed)	0.024	0.968	27.31	0.429	0.713	0.565	0.190
Portal with normaliser	0.097	0.717	4.93	0.227	0.644	0.410	0.082
TF IDF words	0.410	0.243	0.39	0.441	0.787	0.590	0.275
TF IDF characters	0.437	0.289	0.44	0.477	0.797	0.618	0.318
BM25	0.353	0.348	0.66	0.487	0.817	0.626	0.300
MiniLM embeddings	0.360	0.288	0.54	0.455	0.781	0.598	0.287
Learned combiner	0.771	0.684	0.59	0.869	0.985	0.920	0.780

Figure 11 and Table 13 show why raising the threshold does not fix the rule. Precision stays below 7 percent up to a threshold of 90, because so many unrelated pairs reach high scores, while recall falls steadily. Only at 95 and 100, where nearly every component must agree, does precision rise to 28 and 36 percent, by which point recall has fallen to about half. No threshold gives both useful precision and useful recall.

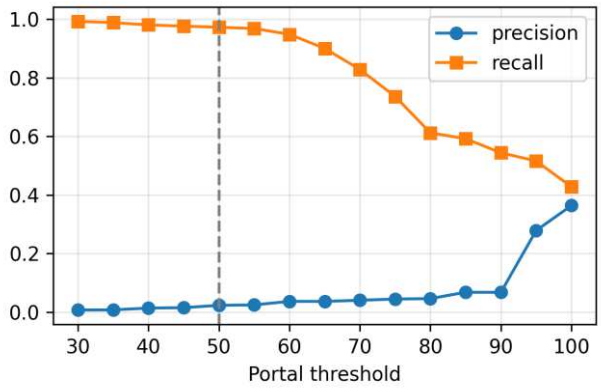

Figure 11. Precision and recall of the deployed rule at thresholds from 30 to 100, medium noise. The dashed line marks the deployed threshold of 50.

Table 13. Deployed rule at different thresholds, medium noise

Threshold	Precision	Recall	F1	Alerts per lost
30	0.008	0.992	0.015	85.00
40	0.014	0.980	0.028	45.36
50	0.024	0.972	0.046	27.29
60	0.037	0.948	0.072	16.98
70	0.041	0.828	0.078	13.48
80	0.047	0.612	0.087	8.75
90	0.068	0.544	0.121	5.33
95	0.279	0.516	0.362	1.23
100	0.364	0.428	0.393	0.78

7.2 Which parts of the rule matter

Table 14 removes one component at a time. Removing the item name points reduced alerts from 27.3 to 15.6 per lost report and raised the share of true items ranked first from 44 to 60 percent. The name component, as implemented with containment, adds more noise than signal: a single shared word such as phone, keys or card is enough for 35 points. Removing the date points had the opposite effect, cutting R1 from 44 to 28 percent; the date is the component that best separates the true item from other items of the same type. Removing the description points cost the most recall, from 97 to 85 percent. The category gate and the location points changed little, the first because most pairs that pass it share a category anyway, the second because owners often name a different or vaguer place than finders.

Table 14. Deployed rule with one component removed, medium noise

Removed	Prec.	Recall	Alerts	R1	MRR
nothing	0.024	0.972	27.29	0.444	0.584
category gate	0.021	0.976	30.44	0.460	0.589
item name	0.038	0.892	15.55	0.604	0.707
description	0.034	0.848	16.70	0.468	0.588
location	0.025	0.960	25.96	0.460	0.576
date	0.025	0.960	25.58	0.284	0.431

7.3 Why the rule misses true matches

Only 7 of the 250 true pairs, 2.8 percent, scored below 50. Table 15 shows what distinguished them. In six of the seven, the item names shared too few words, and in five the categories differed, which cuts the starting points from 35 to 10 or to zero. The same conditions are common among all true pairs: in 12 percent the owner and finder chose different categories, in 30 percent the names shared few words and in 46 percent the location words differed. The rule tolerates these differences only because it awards

points so readily elsewhere, which is also why it raises so many false alerts.

Table 15. Conditions among true matches missed by the deployed rule, medium noise

Condition	Share of missed	Share of all true
categories differ	0.714	0.124
item names share few words	0.857	0.304
location words differ	0.714	0.464
descriptions share few words	0.571	0.056
scored below 50	1.000	0.028

7.4 Comparison of methods across noise levels

The learned combiner was the best method by every ranking measure at every noise level (Table 12, Table 16, Figure 12). At medium noise it ranked the true item first for 87 percent of lost reports and within the top five for 98.5 percent. At its best F1 threshold it sent 0.59 alerts per lost report, 77 percent of them true, and found 68 percent of true matches. At low noise its R1 was 94 percent and at high noise 76 percent, against 47 and 32 percent for the deployed rule.

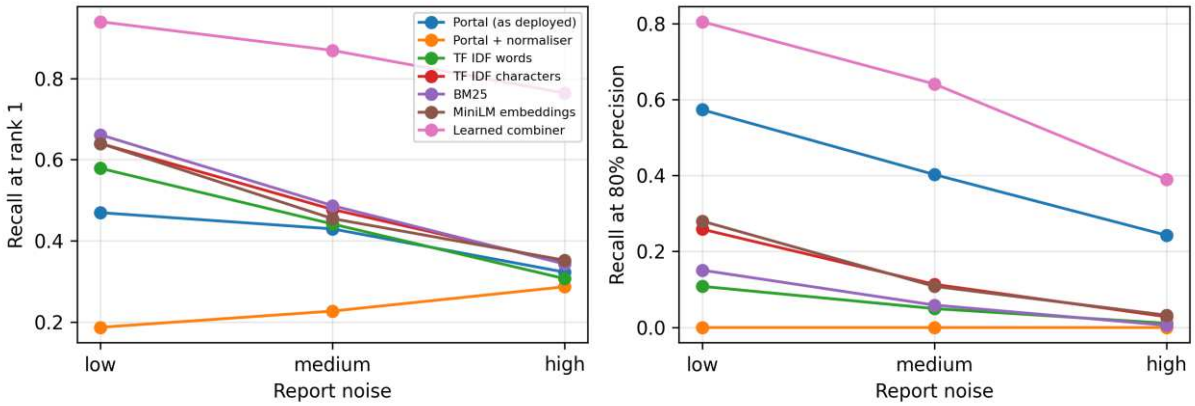

Figure 12. Share of true items ranked first (left) and recall at the validation threshold for 80 percent precision (right) at three noise levels. For the portal the 80 percent threshold (100) reached only 31 to 36 percent precision on the test collections; the normaliser reached 80 percent at no threshold.

Table 16. Ranking quality by noise level, mean of three collections

	R1			AP		
Method	low	med.	high	low	med.	high
Portal	0.469	0.429	0.323	0.213	0.190	0.142
Portal norm.	0.187	0.227	0.287	0.065	0.082	0.091
TF IDF words	0.579	0.441	0.307	0.368	0.275	0.146
TF IDF chars	0.640	0.477	0.347	0.468	0.318	0.182
BM25	0.661	0.487	0.343	0.437	0.300	0.169
MiniLM	0.640	0.455	0.352	0.465	0.287	0.160
Learned	0.940	0.869	0.764	0.868	0.780	0.655

The four single text methods performed alike, with R1 between 44 and 49 percent at medium noise. Sentence embeddings did not beat simple TF IDF or BM25. Their strength, recognising that handset and phone mean the same, is outweighed by their weakness here: two black phones from different owners are semantically almost identical, and what separates them is a brand, a mark, a place or a date, details that embeddings of whole reports blur. The learned combiner succeeds because it keeps these fields apart and learns, from examples, how much agreement on each one is worth.

The normaliser made the rule worse, not better. By mapping synonyms, colours and place aliases to one form, it raised the overlap of unrelated reports as much as that of true pairs, and because the rule awards fixed points, many more pairs reached the maximum. Its R1 fell to 23 percent at medium noise. Its scores improved slightly as noise rose, because noisier reports share fewer words. This shows that the weakness of the rule lies in its point scheme, not in its text cleaning.

Table 17 reports performance when precision is the priority. With thresholds set on validation data for 80 percent precision, the learned combiner kept 64 percent recall at medium noise with 81 percent precision on the test collections. The single text methods kept at most 11 percent recall. The portal's highest threshold reached only 34 percent precision, so the portal cannot be operated at high precision at all.

Table 17. Recall and achieved precision at the validation threshold for 80 percent precision

	Low		Medium		High	
Method	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.
Portal	0.573	0.307	0.403	0.342	0.243	0.357
TF IDF words	0.108	0.798	0.049	0.747	0.011	0.690
TF IDF chars	0.259	0.730	0.113	0.732	0.028	0.661
BM25	0.151	0.731	0.059	0.719	0.005	0.528
MiniLM	0.280	0.699	0.108	0.702	0.032	0.607
Learned	0.805	0.791	0.641	0.807	0.389	0.777

7.5 Stability across test collections

Table 18 shows the range of results over the three medium noise test collections. The learned combiner varied least, with R1 between 0.856 and 0.876, and its lowest result was far above the highest of every other method. The portal's alert count was stable at 27 to 28 per lost report, while its R1 ranged from 0.37 to 0.49 because of the many tied scores.

Table 18. Lowest and highest result over three medium noise test collections

Method	R1	AP	Alerts per lost
Portal	0.372 to 0.488	0.181 to 0.204	26.97 to 27.68
Portal norm.	0.204 to 0.252	0.078 to 0.083	4.86 to 5.01
TF IDF words	0.348 to 0.488	0.232 to 0.309	0.38 to 0.41
TF IDF chars	0.428 to 0.508	0.269 to 0.346	0.40 to 0.50
BM25	0.420 to 0.528	0.244 to 0.330	0.62 to 0.69
MiniLM	0.412 to 0.484	0.270 to 0.305	0.48 to 0.57
Learned	0.856 to 0.876	0.770 to 0.786	0.56 to 0.62

7.6 False alerts grow with the number of open reports

Figure 13 and Table 19 show what happens as reports accumulate. With 100 open found reports, the deployed rule sent 7.4 alerts per lost report, 13 percent of them true. With 2,000 open found reports, the number a busy university could hold over a session if reports are not closed, it sent 151 alerts per lost report, of which 0.7 percent were true. False alerts rose roughly in proportion to the number of open reports, because every new report of a common item type adds another pair that passes the threshold. The normalised rule grew faster still. A matching method whose false alerts grow with the size of the collection becomes unusable exactly when the portal becomes popular.

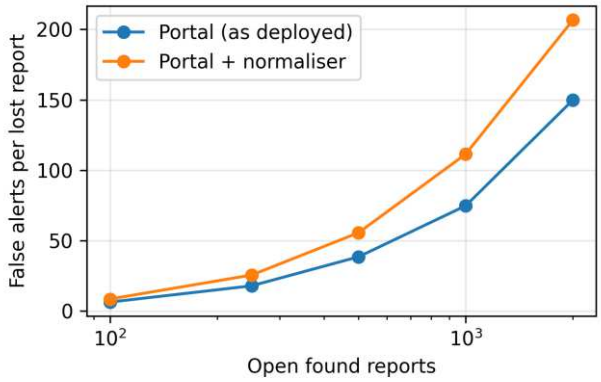

Figure 13. False alerts per lost report as the number of open found reports grows, deployed rule and normalised rule, medium noise (logarithmic horizontal axis).

Table 19. Deployed and normalised rule as open found reports grow

Open found	Method	Precision	Recall	False alerts
100	portal	0.133	0.98	6.40
100	normalised	0.105	1.00	8.56
250	portal	0.051	0.97	17.93
250	normalised	0.037	0.99	25.58
500	portal	0.025	0.97	38.56
500	normalised	0.018	1.00	55.66
1,000	portal	0.013	0.97	74.82
1,000	normalised	0.009	1.00	111.50
2,000	portal	0.007	0.99	149.82
2,000	normalised	0.005	0.99	206.58

7.7 Claim security

The match threshold, which decides which found items a student may claim, offers no protection. A fake lost report written by copying the name, category and description of a public listing passed the threshold for every one of the 250 found items tested. Any student can therefore place any found item in their claim list.

The protection must then come from the administrator's comparison of the claimant's proof with the private fields. Figure 14 and Tables 20 and 21 show how well that works. Genuine owners' proof had a mean similarity of 0.63 to the private fields, with 21 percent matching them completely and 8 percent sharing no words at all, usually because the finder noted no marks or the owner recalled a different place. Impostors limited to the public listing reached a mean of 0.28, and 39 percent shared no words. Proof similarity separated them from owners with a ROC AUC of 0.81. If an administrator accepts claims at a level that admits 80 percent of genuine owners, 37 percent of these impostors are also accepted; at a level that admits about half of owners, 10 percent are. Knowledge of item details is a weak secret, as studies of secret questions have found [1, 24]: common items carry common marks that can be guessed, and genuine owners do not always remember what a finder notices.

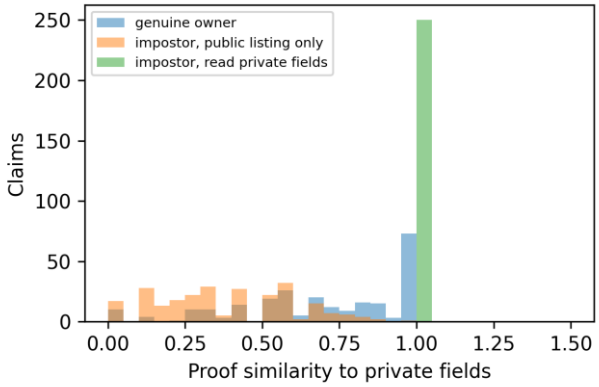

Figure 14. Similarity of claim proof to the found report's private fields for genuine owners, impostors who see only the public listing, and impostors who read the private fields (250 items, portal text similarity).

Table 20. Separating genuine owners from impostors by proof similarity (250 items). The last column applies to both impostor types.

Impostor	Similarity	AUC	Owner mean	Impostor mean	Impostors accepted (50% owners)	Impostors accepted (80% owners)	Copied report passes threshold
public listing only	portal	0.808	0.629	0.277	0.100	0.368	1.00
public listing only	normalised	0.803	0.689	0.379	0.080	0.396	1.00
read private fields	portal	0.106	0.629	1.000	1.000	1.000	1.00
read private fields	normalised	0.146	0.689	1.000	1.000	1.000	1.00

Table 21. Distribution of proof similarity to private fields (portal similarity, 250 items)

Claimant	Mean	Full match	No overlap
Genuine owner	0.629	0.212	0.080
Impostor, public listing	0.277	0.004	0.388
Impostor, read private fields	1.000	1.000	0.000

The impostor who reads the private fields from the database, which the deployed rules permit, defeats the check completely. Their proof matched the private fields exactly in all 250 cases, a better match than most genuine owners achieve, so an administrator comparing the columns of Figure 9 would see the impostor's claim as the stronger one. The ROC AUC of 0.11 means that, by this measure, the impostor looks more like the owner than the owner does. While the private fields remain readable, the claim process verifies nothing.

7.8 Revised matching rule

Table 22 compares the revised public field rule with the deployed rule on the same nine test collections. At medium noise the revised rule sent 0.72 alerts per lost report instead of 27.3, and 39 percent of its alerts were correct instead of 2.4 percent. It ranked the true item first for 49 percent of lost reports and within the top five for 81 percent, against 41 and 72 percent for the deployed rule scored on the same collections. Its alert recall, 42 percent, is lower than the deployed rule's 97 percent, but the owner sees the top five candidates on the dashboard, so most true items remain one click away while the flood of false alerts is gone. The revised rule does not reach the learned combiner, which also used private fields and was trained on labelled pairs; it is the best of the changes that can run safely in the browser today, and it should be replaced by a trained model once real outcomes are recorded. Under the revised rule, a report copied from a public listing still ranks that item first, so the claim list cannot serve as a barrier; ownership must be judged by the administrator from details the public listing does not reveal.

Table 22. Revised public field rule against the deployed rule, mean of three collections per noise level

Noise	Rule	Prec.	Recall	Alerts	R1	R5
low	deployed	0.021	0.983	31.41	0.451	0.793
low	revised	0.422	0.507	0.80	0.549	0.869
medium	deployed	0.024	0.968	27.31	0.413	0.715
medium	revised	0.392	0.424	0.72	0.488	0.807
high	deployed	0.026	0.869	22.64	0.324	0.611
high	revised	0.354	0.323	0.61	0.407	0.720

8 Discussion

8.1 Recommendations for the portal

Keep private fields out of public documents. The revision in Section 5.3 does this with collection level rules and public field matching. When the university can fund a paid plan or a small server, matching can move to a server function with administrator privileges, which would allow the private fields and a trained model to be used for ranking without exposing them to browsers.

Rank instead of alerting on a threshold. The deployed rule cannot be tuned to give few, accurate alerts. A better design shows each owner a short ranked list of the most likely found items and sends an alert only when the top candidate is clearly ahead of the rest. With the learned combiner, the true item was in the top five for 98.5 percent of lost reports at medium noise.

Score fields separately and learn the weights. If a rule must be kept, the evidence suggests removing containment from the item name comparison, giving more weight to date proximity and to distinctive tokens such as brands, names on identity cards and course codes, and treating the category as evidence rather than a gate, since owners and finders disagree on category for about one item in eight. Better still, as reports and outcomes accumulate, administrator decisions provide labelled pairs from which a combiner like ours can be trained on real data.

Do not treat the match score as proof. Because a copied report always passes the threshold, the score should not appear on the claim card as evidence for the claim. Administrators should instead ask claimants for one or two details the finder recorded and the public listing does not reveal, such as a mark, the contents or the colour of a case, and should be told that common items carry common marks. For valuable items, a physical check at the custody location, a photograph of the claimant using the item before the loss, or a phone that can be unlocked are stronger proofs.

Close reports. Because false alerts grow with the number of open reports, found reports that remain unclaimed after a set period, for example one semester, should be closed and moved to a disposal list, and owners should be reminded to close lost reports once the item is recovered.

8.2 Implications for other campus systems

The systems reviewed in Section 2 share the portal's design: a reporting form, a search page, a match notification and an administrator check. Our results suggest three questions that any such system should answer before deployment: how many alerts per report it sends and how many of them are right; whether its private verification fields are protected by the database and not only by the interface; and whether a claim can be made using only public information. Each can be checked with the kind of test collection and impostor simulation used here, at little cost.

8.3 Limitations

The test collections were generated from a catalogue of items, places and writing patterns, not drawn from real UNIPORT reports, which do not yet exist in digital form. Real reports will contain variations we did not model, such as pidgin English, abbreviations unique to particular faculties and photographs, which the portal stores but does not use for matching. The relative ranking of methods is more reliable than the exact numbers. The learned combiner was trained on reports from the same generator as the test collections, so its advantage on real data may be smaller until it is retrained on real outcomes. The impostor models are simple: a determined impostor might do better than our public listing impostor by asking around or watching where items are handed in. The security review is based on the rules and code published in the project documentation; we did not access the live database. Finally, we did not run a usability study with students and administrators.

8.4 Ethics

All reports in the test collections are synthetic; names on simulated identity cards are random combinations of common first names and surnames. Screenshots show test accounts created by the developers. The security findings concern the authors' own system and were shared with the developer, a coauthor, together with the revision in Section 5.3.

9 Conclusion

The Campus Lost and Found Portal replaces UNIPORT's scattered logbooks, notice boards and chat groups with one web application for reporting, searching, matching and claiming lost items, built at low cost on Firebase and Cloudinary. Testing its two central features against known outcomes showed that both need redesign before the portal is scaled up. The matching rule found almost every true match but sent about 27 alerts per lost report, fewer than 3 percent of them correct, and its false alerts grew in proportion to the number of open reports. A learned combination of field similarities ranked the true item first for 87 percent of lost reports against 43 percent for the rule and sent fewer than one alert per report, while single text methods, including sentence embeddings, did no better than about 49 percent. On the security side, the database rules exposed the private details on which claim verification depends, a copied report placed any item in an impostor's claim list, and even without access to private fields, item descriptions separated owners from impostors only moderately. The proposed revision keeps private fields in owner and administrator only collections, ties match suggestions and notifications to real reports, and ranks candidates with a public field rule that cut alerts from 27 to under one per lost report while raising their precision from 2.4 to 39 percent. Future work should collect real reports and outcomes from a pilot at UNIPORT, retrain the matching model on them, add image similarity for reports with photographs, and evaluate the portal with students and security staff.

Data and Code Availability

The experiment is a single Python script that generates the test collections, runs all matching methods and the claim experiment, and writes every table and figure in this paper; it runs in one Kaggle notebook cell with fixed random seeds. The script, results and portal source code are available from the authors.

References

[1] Joseph Bonneau, Elie Bursztein, Ilan Caron, Rob Jackson, and Mike Williamson. 2015. Secrets, Lies, and Account Recovery: Lessons from the Use of Personal Knowledge Questions at Google. In *Proceedings of the 24th International Conference on World Wide Web (WWW 2015)*. ACM, 141–150. doi:10.1145/2736277.2741691

[2] Andrei Z. Broder. 1997. On the Resemblance and Containment of Documents. In *Proceedings of Compression and Complexity of Sequences 1997*. IEEE, 21–29. doi:10.1109/SEQUEN.1997.666900

[3] Peter Christen. 2012. Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection. Springer, Berlin. doi:10.1007/978-3-642-31164-2

[4] N. M. David, G. M. Catapang, K. P. V. Pabilando, L. T. Dala, R. F. Marcoso, and A. D. Lacasandile. 2024. A Comparative Study on Lost and Found Management Systems in Academic Institutions. In *Proceedings of the 6th International Workshop on Artificial Intelligence and Education (WAIE 2024)*. IEEE, 230–236. doi:10.1109/WAIE63876.2024.00049

[5] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. 2007. Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering* 19, 1 (2007), 1–16. doi:10.1109/TKDE.2007.250581

[6] Ramez Elmasri and Shamkant B. Navathe. 2016. Fundamentals of Database Systems. Pearson, Hoboken, NJ.

[7] Federal Republic of Nigeria. 2023. Nigeria Data Protection Act, 2023. Official Gazette, Abuja.

[8] Ivan P. Fellegi and Alan B. Sunter. 1969. A Theory for Record Linkage. *Journal of the American Statistical Association* 64, 328 (1969), 1183–1210. doi:10.1080/01621459.1969.10501049

[9] Jerome H. Friedman. 2001. Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics* 29, 5 (2001), 1189–1232. doi:10.1214/aos/1013203451

[10] Google. 2026. Firebase Documentation: Cloud Firestore Security Rules, Securely Query Data and Rules Are Not Filters. https://firebase.google.com/docs/firestore/security/rules-query.

[11] Mordechai Haklay and Patrick Weber. 2008. OpenStreetMap: User Generated Street Maps. *IEEE Pervasive Computing* 7, 4 (2008), 12–18. doi:10.1109/MPRV.2008.80

[12] Paul Jaccard. 1912. The Distribution of the Flora in the Alpine Zone. *New Phytologist* 11, 2 (1912), 37–50. doi:10.1111/j.1469-8137.1912.tb05611.x

[13] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. Introduction to Information Retrieval. Cambridge University Press, Cambridge, UK.

[14] Peter Mell and Timothy Grance. 2011. The NIST Definition of Cloud Computing. doi:10.6028/NIST.SP.800-145

[15] Vimalnath Murugiah and Rosmamalmi Mat Nawi. 2025. FindItHere: Lost and Found Item Matching Web Application. *Applied Information Technology and Computer Science* 6, 2 (2025), 453–472.

[16] Gonzalo Navarro. 2001. A Guided Tour to Approximate String Matching. *ACM Computing Surveys* 33, 1 (2001), 31–88. doi:10.1145/375360.375365

[17] OWASP Foundation. 2021. OWASP Top 10: 2021, A01 Broken Access Control. https://owasp.org/Top10/A01_2021-Broken_Access_Control/.

[18] D. Pede, V. Ranawat, R. Shete, S. Gole, and T. Kolhe. 2025. Lost and Found Web Application. *International Journal of Innovative Science and Research Technology* 10, 5 (2025), 1242–1244. doi:10.38124/ijisrt/25may811

[19] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[20] Jennifer Preece, Yvonne Rogers, and Helen Sharp. 2019. Interaction Design: Beyond Human Computer Interaction. John Wiley and Sons, Indianapolis, IN.

[21] Nils Reimers and Iryna Gurevych. 2019. Sentence BERT: Sentence Embeddings Using Siamese BERT Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language*

Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP IJCNLP). Association for Computational Linguistics, 3982–3992. doi:10.18653/v1/D19-1410

[22] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval* 3, 4 (2009), 333–389. doi:10.1561/1500000019

[23] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. 1996. Role Based Access Control Models. *IEEE Computer* 29, 2 (1996), 38–47. doi:10.1109/2.485845

[24] Stuart Schechter, A. J. Bernheim Brush, and Serge Egelman. 2009. It's No Secret: Measuring the Security and Reliability of Authentication via Secret Questions. In *Proceedings of the 30th IEEE Symposium on Security and Privacy*. IEEE, 375–390. doi:10.1109/SP.2009.11

[25] Nurul Fatihah Shahzan and Nureize Arbaiy. 2025. UTHM Lost and Found Management System. *Applied Information Technology and Computer Science* 6, 1 (2025), 2307–2319.

[26] Ian Sommerville. 2016. Software Engineering. Pearson, Boston, MA.

[27] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. MiniLM: Deep Self Attention Distillation for Task Agnostic Compression of Pre Trained Transformers. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, 5776–5788.