

Development of a Web-based System for Matching Losers and Finders of Personal Items

Ajibola O. Oyedeji*, Olaolu Folorunsho, Olatilewa R. Abolade and

Nicholas I. Eigbiremonlen

Department of Computer Engineering

Olabisi Onabanjo University

Ago-Iwoye, Nigeria

*oyedeji.ajibola@oouagoiwoye.edu.ng

Date received: February 15, 2020

Revision accepted: May 6, 2021

Abstract

A considerable amount of time is wasted in the process of searching for a missing item and consequently, human schedules are disrupted. This study aimed to design and implement a web-based system for matching losers and finders of personal items using the waterfall model. The modelling language employed in this work was Unified Modeling Language (UML). The server-side was built using the Node.js JavaScript framework to provide backend logic to process information from the client-side and MongoDB database. The client-side was made using Hypertext Markup Language (HTML), cascading style sheet (CSS) and JavaScript. The system was evaluated using alpha and beta testing by module testing and getting feedback from users based on some predefined criteria. The developed system was able to match lost items with efficiency and functionality of 80% and 77%, respectively.

Keywords: *UML, waterfall model, programming, matching losers and finders*

1. Introduction

As the world gets busier over time, the chance of missing items gets higher. Items may be categorized based on the function or importance. Some items tend to be of higher priority than other items (Ahmad *et al.*, 2014). According to a survey done by the College of Computing and School of Psychology, Georgia Institute of Technology, a significant percentage of the respondents admitted that finding lost objects is a concern of everyday life (Pak *et al.*, 2004). The need to have these items available always, especially when they are needed, makes losing them unpleasant. Ahmad *et al.* (2015) identified time, money and quantity as things that could be affected.

Pak *et al.* (2004) dived into giving massive information about the factors that influence finding objects. The paper investigated the real-world nature of what losing an object means and the strategies used to find them. Accordingly, there were common strategies of people in finding their missing items regardless of age or object type. They were grouped into retracing, memory, delegated, locus and exhaustive search. The most frequently employed strategies were locus search (33%), followed by exhaustive (24%) and retracing search (19%). The remaining strategies, memory and delegated search, were both reported 11% of the time (Srinivasan *et al.*, 2014).

Srinivasan *et al.* (2014) developed a tracking device to track items when missing. The device built used Bluetooth and global positioning system (GPS) technologies and the development of a mobile application. The system involves the connection of all items that can be misplaced to a wireless network (Bluetooth), which results in increased cost of implementation and deployment. It is particularly inefficient as any unconnected device cannot be searched using the developed system. Alnaghaimshi *et al.* (2020) made a smartphone application that solves the missing item problem in Arabic. By relying on the contribution of people in online social networks, the idea of this project was to help people in finding the missing objects by creating an Arabic online social community (Alnaghaimshi *et al.*, 2020). A prototype web-based application for effective online reporting and searching of lost and found items in a crowd gathering was implemented for visitors to Hajj in Makkah and Madinah (Nadeem *et al.*, 2020).

Furthermore, the study conducted by Chan *et al.* (2009) proposed a mobile device that utilizes radio frequency identification (RFID) and GPS to keep track of tagged items' locations. The proposed RFID-based system sends an alert to the user once the item is out of range of the RFID reader. The system requires the user to receive the alert in time and the RFID-based systems cannot keep track of the lost items exceeding a range of 1.5 m. A mobile crowdsourced guiding system, codenamed EasyFind, which finds lost or missing items and locates people by utilizing the computing power of the internet of things (IoT), was designed, developed and deployed (Chen and Liu 2019). The system locates lost items that have been preinstalled with mobile iBeacon nodes through ad-hoc sensing networks created by users with smartphones in the vicinity. The system works well within a set location or distance.

In all these systems, it is necessary to locate quickly some or all occurrences of user-specified words or phrases in one or several arbitrary text strings. From online matchmaking and dating sites to medical residency placement programs, matching algorithms are used in areas of spanning, scheduling, planning, pairing of vertices and network flows. String-matching algorithms are basic components used in implementations of practical software existing under most operating systems where a text is looked up in a database or index (Knuth *et al.*, 1977; Hall and Dowling, 1980; Karp and Rabin, 1987).

In the school context, there is a need for an easy-to-use, inexpensive campus finder system due to the high rate of missing and lost items as well as the high cost implementation of the existing systems. Therefore, this study aimed to design and implement a web-based system for matching losers and finders of personal items that would result in reduced stress and time associated with locating lost personal items. The developed system requires no additional hardware such as electronic tags or beacons; hence, the low-cost implementation and deployment. Also, the system is not limited by distance in effectively locating the lost items. Since it is web-based, the finders system only requires internet access.

2. Methodology

The type of software development life cycle model utilized in this study is waterfall model because it is flexible to use (Oyediji *et al.*, 2019). The process is shown in Figure 1 below.

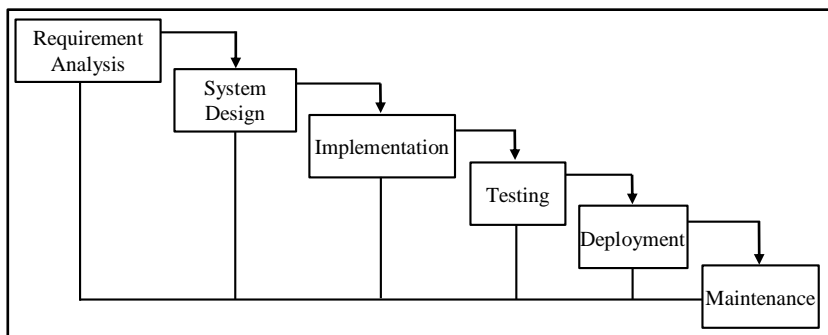


Figure 1. Waterfall model

2.1 System Requirements Analysis

The first step in software development is system requirement gathering. The authors brainstormed to determine the functional and non-functional requirements of the system. The requirement questions are the following:

Who: This question addresses the parties involved in the usage of this system. This includes every student who could lose an item or find a missing item on behalf of the owner.

What: What is the system used and what problem is it solving? This system is meant to get a description of items either from the finder or the owner of the item, match the information from both ends, conclude and return the item that matches the missing item in the database.

How: How is the system going to solve the problem? Some technology will work to get input from the user. Others will handle the logic that goes behind the scene to see to the working of the system.

The system was designed to work as a web-based application with an administrator and verified users who are members of the university. The requirements of the system are 1) users' authentication details for the admin; 2) user interface to show and take in information from users of the platform and interact with the data source; 3) database design and implementation for data storage and retrieval; and 4) string-matching algorithm to match the string entered by the user against the information available in the database.

2.2 System Design

The functional modules of the system were designed and the different interconnections or flow of information between the modules were clearly stated. The design of the system was made by using object-oriented analysis and design and graphical user interface (GUI) for user's easy interaction with the system. The design consisted of different components including a user interface.

2.2.1 Class Diagram

The class diagram describes the static structure of the system at the classifier level as shown in Figure 2. It indicates the class names, attributes and operations of the system and their class relationship. The user class has a one-

to-many relationship with the item class. The attributes are indicated with the userItem being updated with a list of item attached to the owner. The user class is a superclass for ItemFinder and ItemOwner; this implicates that the two classes are a specific class of users.

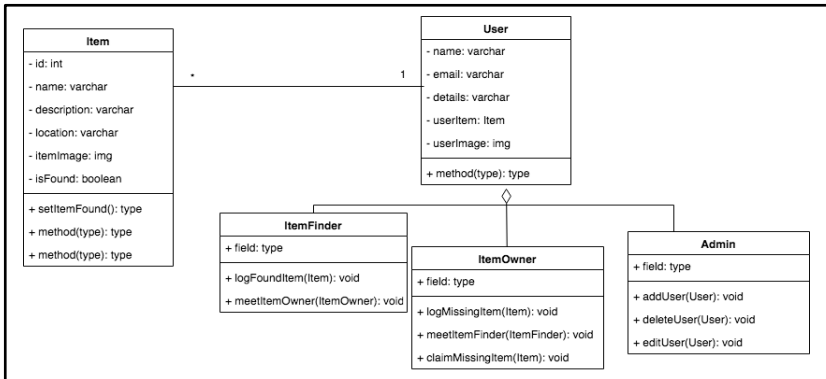


Figure 2. Class diagram of the matching system

2.2.2 Activity Diagram

The activity diagram shows the system's activity flow as presented in Figure 3.

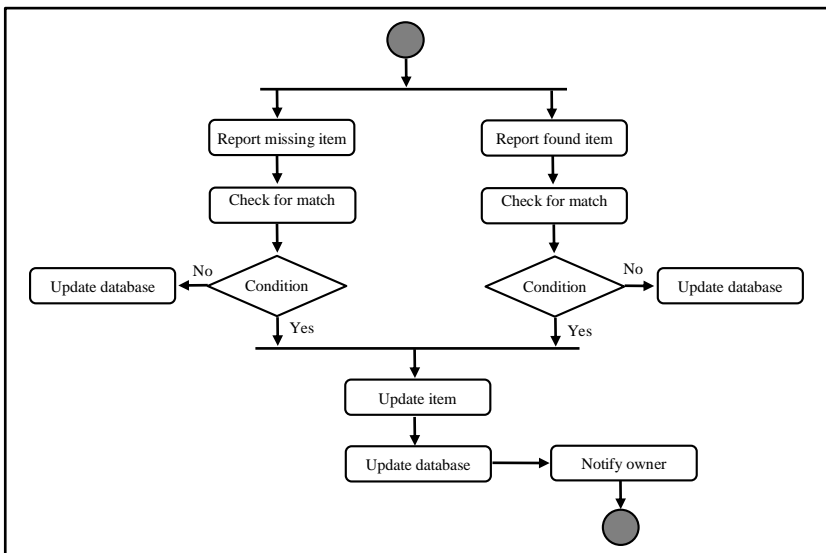


Figure 3. Activity flowchart of the matching system

Initially, the user is authenticated either by creating a new user or by validating the details of existing users. The user could then report a missing or found item. In both cases, the reported item would be run through the database to see matches with existing alternative; missing item will run through a database of previously reported found item while the found item will run through a database of the previously reported lost item.

2.2.3 Use Case Diagram

Figure 4 presents the use case diagram which was used to design the functional requirements of the system including the login page, profile page and adding or deleting of items. The user can login, update profile and add an item on the system as either lost or found. The admin, on the other hand, can update and delete an item as well as delete a user in case of irregularities.

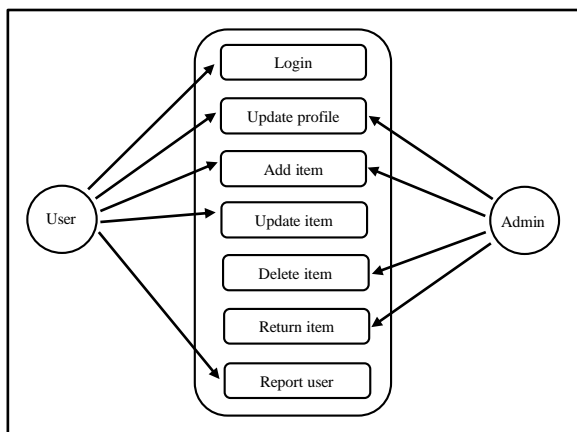


Figure 4. Use case diagram of the matching system

2.2.4 String-Matching Algorithm

A string-matching algorithm was utilized to match the lost item code and the found item code. This type of algorithm is useful in the case of searching a string within another string for applications such as database schema and network systems. String-matching algorithms are basic components used in the implementation of practical systems existing under most operating systems. Most exact string pattern-matching algorithms are easily adapted to deal with multiple string pattern searches or with wildcards. Exact string-matching consists of finding one or more, generally, all of the occurrences of a pattern in a target.

For this system, the Knuth-Morris-Pratt (KMP) algorithm, which is an exact string-matching algorithm, was implemented. The string-matching algorithm of the web application for matching losers and items' finders was implemented by checking users' input with the data in the database using the KMP algorithm shown in Figure 5. The system searches the user input against the records of the reported lost and found items in the database and returns a match report if found.

```

algorithm kmp_search:
  input:
    an array of characters, S (the text to be searched)
    an array of characters, W (the word sought)
  output:
    an array of integers, P (positions in S at which W is found)
    an integer, nP (number of positions)

  define variables:
    an integer, j  $\leftarrow$  0 (the position of the current character in S)
    an integer, k  $\leftarrow$  0 (the position of the current character in W)
    an array of integers, T (the table, computed elsewhere)

  let nP  $\leftarrow$  0

  while j < length(S) do
    if W[k] = S[j] then
      let j  $\leftarrow$  j + 1
      let k  $\leftarrow$  k + 1
      if k = length(W) then
        (occurrence found, if only first occurrence is needed,
        m  $\leftarrow$  j - k may be returned here)
        let P[nP]  $\leftarrow$  j - k, nP  $\leftarrow$  nP + 1
        let k  $\leftarrow$  T[k] (T[length(W)] can't be -1)
      else
        let k  $\leftarrow$  T[k]
        if k < 0 then
          let j  $\leftarrow$  j + 1
          let k  $\leftarrow$  k + 1

```

Figure 5. KMP algorithm

2.3 Implementation of the System

The programming codes were written using the Visual Studio Code (VS Code) integrated development environment (IDE), which includes features with support for debugging, embedded Git control, syntax highlighting, intelligent code completion, snippets and code refactoring.

The client-side of the implementation was built using Hypertext Markup Language (HTML), cascading style sheet (CSS), and JavaScript. The server-

side was made utilizing Node.js on this software (VS Code). The inbuilt code auto-completion aided ease of writing code in the IDE. Git was also utilized from the GUI provided by IDE and inbuilt terminal.

The system's database used MongoDB, which is a free and open-source cross-platform document-oriented database program, for storing information such as the items reported by the users and admin information. This was done through mongoose driver – a Node Package Manager (NPM) module that can be used in a Node.js application. Node.js was employed in writing the server-side's backend logic and implementing the mongoose database driver for MongoDB and string-matching module among others. The code snippet responsible for handling the information from the user and responding to the appropriate page or information was implemented using the Express Node.js Framework.

2.4 Matching System Testing and Evaluation

The system was evaluated using alpha and beta testing to ensure the system developed meets its specification requirement.

2.4.1 Alpha Testing

Alpha testing was carried out during every phase of the system development process to ensure that all the modules function as expected. This included functional and non-functional testing to verify the performance, reliability and usability of the system.

2.4.2 Beta Testing

Beta testing was done and performed by potential and real end-users to test and rate the system based on criteria such as clarity, attractiveness, responsiveness and orthography. A total of 10 respondents, who are students of the Department of Computer Engineering of Olabisi Onabanjo University, Nigeria, tested the developed system. The testing was based on clarity (ease of use of the system), user interface (level of friendliness of the user interface), functionality (correctness of the system in matching lost and found items) and maintainability (ease of updating and including more functions). The evaluation was done on a scale of 1 to 10 – 1 as poor and 10 representing excellent.

3. Results and Discussion

3.1 User Interface Design

3.1.1 User Section

The homepage of the user interface, usually known as the index page, presents the user with a welcome message and the necessary buttons that navigate to a form used for reporting missing items as shown in Figure 6. On the front page, a form to help the users in making a follow-up on their reported item was included. A contact-us form is added to the homepage (at the tail end of the page) for users to contact the admin. The form interface is well-structured to take the input from the users. The user can add details of the item in either found or lost item form.

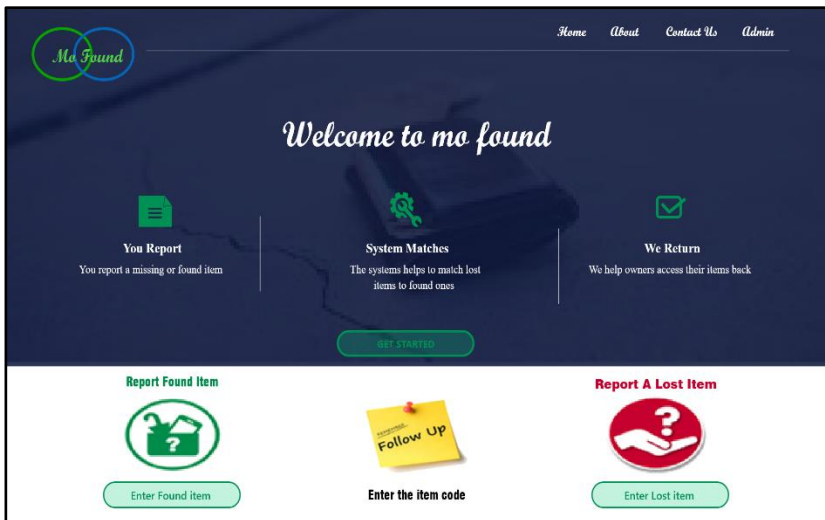



Figure 6. User homepage

There is an item summary page where the information such as the item name and description is displayed to the user who reported the item or has access to the item code. Figures 7 and 8 show the summary page and the information presented to the users. The user is given the option to update and delete the item from this section.



[Home](#) [About](#) [Contact Us](#)

Found School ID Card

Item Code: ElwL3s8

Thanks, the item was registered Successfully, you will be contacted when a match is found. Copy the code above to follow up on your item.
If the cod doesn't load the page, that means the match has been found and returned to the owner.

Report at any of this locations: OAU Senate

Item Name:

ID card

Item Category:

Document

Item Brand:

Unknown

Major Color:

white

Loss Location:

Unknown

Generated Code:

White Unknown ID and My school ID card with the name Nikia Sunday in Document Category

Status:

Not Found


Reporter:


09038727766

DELETE

UPDATE

Share on:

 Facebook

 Twitter



 Whatsapp

Figure 7. Reported found item page



[Home](#) [About](#) [Contact Us](#)

Lost ID Card

Item Code: Brzuoo5

Thanks, the item was registered Successfully, you will be contacted when a match is found. Copy the code above to follow up on your item.
If the cod doesn't load the page, that means the match has been found and returned to the owner.

Report at any of this locations: OAU Senate

Item Name:

ID card

Item Category:

Document

Item Brand:

Unknown

Major Color:

white

Loss Location:

Unknown

Generated Code:

White Unknown ID and My school ID card with the name Nikia Sunday in Document Category

Status:

Not Found


Reporter:


09038727766

DELETE

UPDATE

Share on:

 Facebook

 Twitter


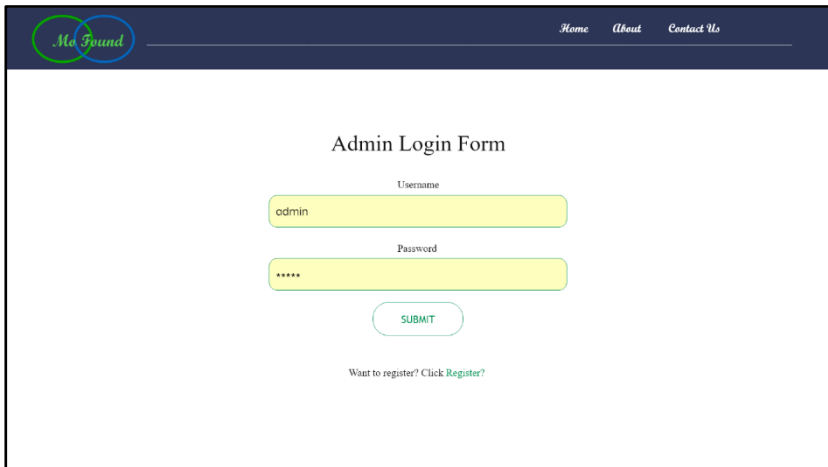
 Whatsapp

Figure 8. Reported lost item page

3.1.2 Admin Section

There are interfaces specific to the system’s admin. In this section, the admin performs activities of the matching system. The admin can create, read, update and delete items using the admin section or dashboard.

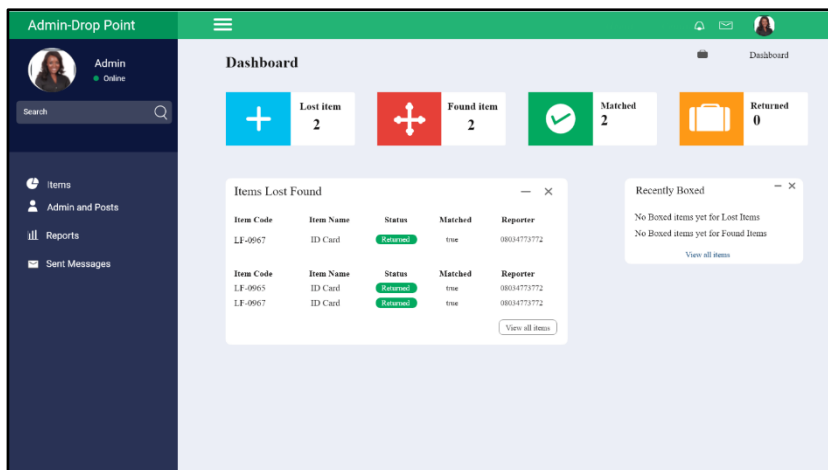
The login page first renders to the admin because entering the admin dashboard requires authentication. The login page (Figure 9) shows the admin trying to login into the admin dashboard.



The image shows a web page titled "Admin Login Form". At the top, there is a navigation bar with a logo "Ma Found" and links for "Home", "About", and "Contact Us". The main content area has a title "Admin Login Form" and two input fields: "Username" with the value "admin" and "Password" with masked characters "*****". Below the password field is a "SUBMIT" button. At the bottom, there is a link "Want to register? Click [Register](#)".

Figure 9. Admin login page

Admin dashboard (Figure 10) renders to an authenticated admin with information such as the list of all reported items, number of reported found, lost, returned and boxed items.



The image shows an "Admin-Drop Point" dashboard. The top navigation bar is green and contains a search icon, a bell icon, an envelope icon, and a user profile icon. The left sidebar is dark blue and contains a search bar, a user profile icon, and links for "Items", "Admin and Posts", "Reports", and "Sent Messages". The main content area is light blue and contains a "Dashboard" section with four cards: "Lost item 2", "Found item 2", "Matched 2", and "Returned 0". Below these cards is a table titled "Items Lost Found" with columns for "Item Code", "Item Name", "Status", "Matched", and "Reporter". The table contains three rows of data. To the right of the table is a "Recently Boxed" section with a "View all items" link.

Item Code	Item Name	Status	Matched	Reporter
LF-0967	ID Card	Returned	true	08034773772
LF-0965	ID Card	Returned	true	08034773772
LF-0967	ID Card	Returned	true	08034773772

Figure 10. Admin dashboard

The admin can also see if an item has been matched yet or not. Each item on the list has a link that navigates to the object’s detail where the admin can update or delete the item. The updated form for the admin expects more information from the admin. The admin has to specify if the item is boxed or not or specify if the item has been returned or not. The admin’s update item form shows the information needed from the admin for item updating.

3.2 Evaluation Result

The evaluation result of the beta testing stage is presented in Table 1. The evaluators were selected randomly to use and report items to be missing or found. The system was able to match four out of the five items (0.8) that were real matches when the testers were told to report in pairs the same item from adjacent report cases. This puts the efficiency of the system at 80%. Based on the results, the clarity of the system obtained the lowest score at 64% while functionality was at the highest (77%).

Table 1. System evaluation result

User	User Interface	Clarity	Functionality	Maintainability
A	7	6	7	5
B	6	6	8	4
C	6	4	9	8
D	9	5	6	9
E	7	7	7	9
F	7	8	8	7
G	6	7	9	9
H	8	5	9	7
I	8	8	6	5
J	5	8	8	7
Average Rating	6.9	6.4	7.7	7
Percentage (%)	69%	64%	77%	70%

4. Conclusion and Recommendation

To find easily and use personal items when they are needed can be said to be of optimum importance to the item owner. A matching system that assists in returning the lost personal items can help ease the process of finding them. The study brings to light a system capable of taking a report from a finder or reporter and report if a match is found or not. The study was able to design

and develop a string-matching system using reports from the losers and finders of personal items with an efficiency of 80%.

The system can be built to serve different communities or organizations specifically in a customized way. It is recommended to build a web service with a customized system that uses general matching algorithm by providing representational state transfer application program interface (REST API) endpoints that can be implemented by different clients in their organizations.

5. References

- Ahmad, S., Lu, R., Zhang, Y., & Ziaullah, M. (2014). An empirical study of smartphone-based tracking devices and analysis of its demand & market. *International Journal of Economics, Commerce and Management*, 2(12), 1-13.
- Ahmad, S., Ziaullah, M., Rauniyar, L., Su, M., & Zhang, Y. (2015). How does matter lost and misplace items issue and its technological solutions in 2015 - A review study. *IOSR Journal of Business and Management*, 17(4), 79-84.
- Alnaghaimshi, N.I., Alenizy, R.A., Alfayez, G.S. & Almutairi, A.A. (2020). Mafqudat: Arabic smartphone application for reporting lost and found items. *Proceedings of the 3rd International Conference on Computer Applications & Information Security (ICCAIS)*, Riyadh, 1-4.
- Chan, S., Connell, A., Madrid, E., Park, D., & Kamoua, R. (2009). RFID for personal asset tracking. *IEEE Long Island Systems, Applications and Technology Conference*, Farmingdale, NY, USA, 1-7.
- Chen, L.W., & Liu J.X. (2019). EasyFind : A mobile crowdsourced guiding system with lost item finding based on IoT technologies. *Proceedings of IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, Kyoto, Japan, 343-345.
- Hall, P.A., & Dowling, G.R. (1980). Approximate string matching. *ACM Computing Surveys*, 12(4), 381-402.
- Karp, R.M., & Rabin, M.O. (1987). Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development*, 31, 249-260.
- Knuth, D. E., Morris, J. H. & Pratt, V. R. (1977). Fast pattern matching in strings. *SIAM Journal on Computing*, 6(2), 323-350.
- Nadeem, A., Rizwan, K., Syed, T.A., Alkhodre, A., & Mehmood, A. (2020). Spatio-temporal modeling and application for efficient online reporting and tracking of lost items during huge crowd gatherings. *International Journal of Computing and Digital Systems*, 9(6), 1156- 1163. <http://dx.doi.org/10.12785/ijcds/0906013>

Oyediji, A.O., Osifeko, M.O., Folorunsho, O., Abolade, O.R., & Ade-Ikuesan, O.O. (2019). Design and implementation of a medical diagnostic expert system. *Journal of Engineering Science*, 10(2), 103-109.

Pak, R., Peters, R.E., Rogers, W.A., Abowd, G.D., & Fisk, A.D. (2004). An analysis of why people lose objects, how they find them, and their attitudes about a technology aid. *Proceedings of the Human Factors and Ergonomics Society 48th Annual Meeting*, New Orleans, USA, 262-265.

Srinivasan, P., Antonia, S., Rekha, A., & Anbarasu, V. (2014). Locate misplaced objects: GPS-GSM-Bluetooth enabled tracking. *Journal of Computer Trends and Technology*, 9(1), 10-13.