International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

Traceit APP-ONE STOP DESTINATION FOR LOST/FOUND ITEM ISSUE.

Prof. Renuka Arbat[1], Nandini Rao[2], Somesh Vernekar[3], Soham Kokate[4]

MIT School of Computing
MIT ADT University
Loni Kalbhor, Pune

Abstract

The growing frequency of misplaced or lost personal belongings has led to a demand for intelligent and accessible tracking solutions. This paper presents TraceIt, a mobile application designed to assist users in locating their lost items through a combination of user tagging, image recognition, and real-time tracking. Built using Flutter and Firebase, TraceIt enables users to upload item details, report losses, and potentially identify recovered items through a crowdsourced platform. The app architecture supports scalability and future enhancements such as Machine Learning (ML) for image analysis and GPS-based location tagging.

1. INTRODUCTION

Losing valuable personal items such as mobile phones, wallets, bags, or keys is a common occurrence that can lead to stress, financial loss, and inefficiency. Traditional lost-and-found systems are often localized, manual, and lack a centralized platform for reporting and retrieving items. To address this gap, TraceIt was conceived as a mobile application that harnesses the power of user participation, digital data storage, and real-time access to help track and recover lost items.

Unlike physical boards or isolated systems in public places, TraceIt offers a centralized, scalable solution accessible from anywhere. This paper explores the comprehensive design and implementation of the application while highlighting its potential to evolve into a robust smart recovery system with the integration of AI and GPS technologies.

2. LITERATURE REVIEW

Existing solutions for tracking lost items range from simple notice boards to sophisticated Bluetooth and GPS-based tracking devices. Products like Apple AirTag and Tile offer real-time tracking, but they require physical tags and are not cost-effective for all users. Academic research has explored IoT-based solutions, including the use of RFID and Arduino-powered devices to create smart tracking systems. However, these require technical knowledge and infrastructure that are not user-friendly for the general public.

In contrast, app-based solutions are more accessible and can leverage crowd-sourced data. Some mobile apps exist for lost-and-found purposes, but they often suffer from low user engagement and limited functionality.

The integration of cloud services like Firebase offers a promising approach due to its scalability, ease of integration, and cost-effectiveness. Image recognition, powered by ML models, has been successfully applied in various domains such as face recognition and object detection, which opens the possibility of using it to match lost items visually. However, such implementations are rare in the lost-and-found domain, presenting an opportunity for innovation.

Mobile app-based solutions are emerging as more accessible alternatives, with several applications designed for lost-and-found purposes. However, many suffer from drawbacks such as poor user interface design, low engagement rates, lack of real-time updates, and the absence of advanced matching algorithms. Few apps capitalize on the power of community participation or integrate features like automatic image recognition.

The integration of cloud platforms such as Firebase has revolutionized mobile development by offering scalable storage, authentication, and real-time database capabilities. Firebase's ease of integration makes it suitable for rapid development and deployment of lost-and-found solutions.

Machine Learning (ML) has demonstrated great success in domains such as facial recognition, medical imaging, and autonomous systems. Image recognition using convolutional neural networks (CNNs) and pretrained models like MobileNet and ResNet opens the door for automated item identification. Although rarely used in lost-and-found contexts, these technologies offer immense potential for intelligent visual matching.

Crowdsourcing and community-based data collection, similar to platforms like Waze or Google Maps contributions, have shown success in increasing data richness and usability. Such approaches, when combined with cloud computing and ML, can significantly improve the efficacy of item recovery applications.

Academic research has investigated IoT-based solutions using RFID, NFC, and Arduino-powered systems to automate item tracking. While technically impressive, these setups demand significant hardware infrastructure, technical knowledge, and maintenance, which are impractical for general public use.

Therefore, the TraceIt application attempts to bridge the gap by providing a scalable, accessible, community-based mobile app that leverages cloud technologies and has the potential to integrate ML and GPS-based tagging in the future.

scalable, user-friendly system for fake news detection in dynamic environments.

3. METHODOLOGY

Development Approach

TraceIt was developed using the Agile software development methodology, which allowed for iterative improvements based on feedback and testing. The development was structured into the following phases:

1. **Requirement Gathering:** Understanding user needs and defining the core functionality.
2. **Design:** Wireframing the app screens, database schema, and system architecture.
3. **Development:** Building the front-end using Flutter and integrating Firebase services.

4. **Testing:** Functional testing, UI testing, and database validation.

5. **Deployment:** Releasing the app for testing on Android.

Tools and Technologies Used

- **Flutter**: For cross-platform mobile app development.
- **Firebase Authentication**: For secure user sign-in.
- **Firebase Firestore**: For storing item metadata.
- **Firebase Cloud Storage**: For storing images of reported items.
- **Firebase Cloud Messaging (future scope)**: For notifying users of matches.

App Architecture and Data Flow

1. **User Login** → 2. **Data Input (Lost/Found Item Report)** → 3. **Image Upload** → 4. **Store in Firestore/Cloud Storage** → 5. **Match Checking & Search** → 6. **Result Display**

User-Centric Design

- Prioritized usability by simplifying forms.
- Enabled category filtering and image thumbnails.
- Designed UI to accommodate diverse age groups and tech literacy levels.

friendly means of inputting news articles and receiving instant authenticity verification. This comprehensive methodology ensures that the fake news detection system is accurate, scalable, and practically deployable in real-world scenarios.

Data Structuring and Query Optimization

- Indexed Firestore queries to support fast search and filtering.
- Structured item documents with timestamps, image URLs, and metadata for future ML compatibility.

Ethical and Security Considerations

- Role-based data access policies using Firestore security rules.
- Privacy-preserving design to avoid overexposing user contact information.

This comprehensive methodology ensures TraceIt is robust, adaptable, and ready for future enhancements like GPS tracking and machine learning integration.

4. SYSTEM DESIGN AND FEATURES

The TraceIt application follows a modular and service-oriented architecture that separates responsibilities across the frontend, backend, data storage, and authentication layers. This design ensures scalability, maintainability, and ease of integration with future enhancements.

Frontend: Flutter Mobile App

- **Cross-Platform Support**: Developed using Flutter, the app supports both Android and iOS from a single codebase, ensuring consistent performance and UI across platforms.
- **User Interface (UI)**: Implements intuitive layouts for login, reporting lost/found items, browsing items, viewing matches, and managing user profiles.
- **State Management**: Uses Provider for efficient state handling and UI updates.
- **Navigation**: Employs Flutter's built-in Navigator for smooth screen transitions and routing.

Backend: Firebase Services

- **Serverless Architecture**: Firebase offers a Backend-as-a-Service (BaaS), eliminating the need for managing traditional server infrastructure.
- **Real-Time Data Handling**: Firebase Firestore provides real-time updates, allowing users to see newly reported items instantly.
- **Cloud Functions (Future Scope)**: Can be integrated for server-side business logic such as automated image processing or data moderation.

Data Storage

- **Firestore Database**:
  - Used to store item metadata such as title, description, category, user ID, date, and status (lost/found).
  - Enables fast querying and filtering of items.
  - Ensures structured storage with support for indexing and real-time synchronization.
- **Firebase Cloud Storage**:
  - Stores high-resolution images of reported items.
  - Generates public or authenticated URLs for image access.

- Secure and scalable for handling large volumes of media.

Authentication: Firebase Auth (Google Login)

- **Google Sign-In**: Enables users to log in securely using their Google accounts, ensuring a seamless onboarding experience.

- **Session Management**: Firebase handles authentication tokens and session renewal automatically.

- **Data Security**: User access to Firestore and Storage is governed by Firebase security rules based on authentication status.

- **User Identity**: Logged-in users have unique identifiers (UIDs) that are used to associate item reports and personalize data.

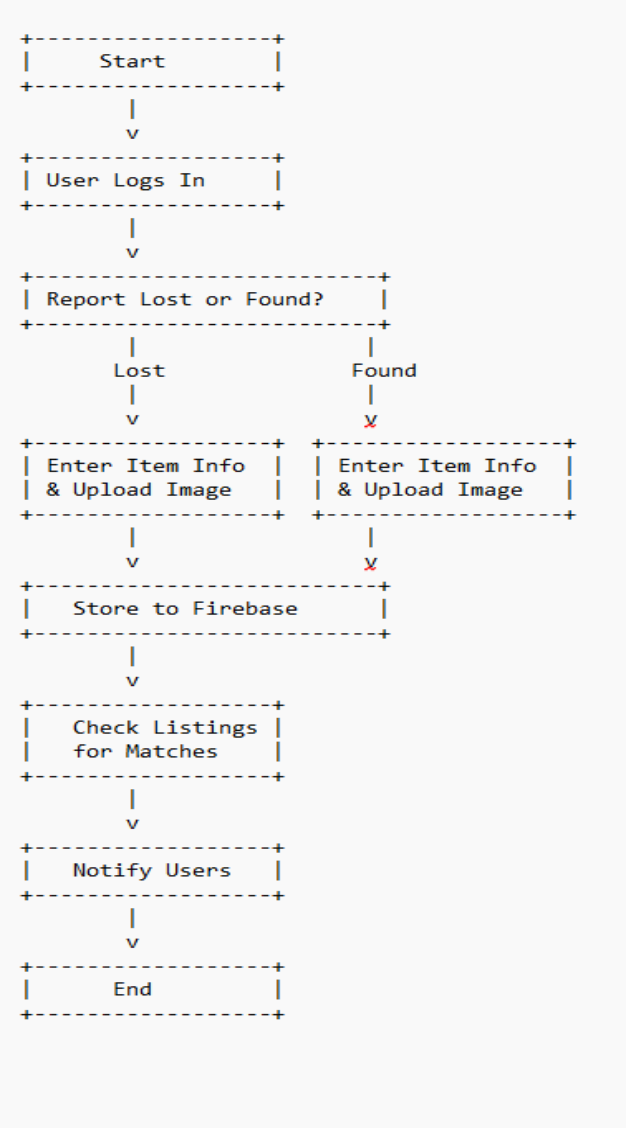

Fig. 1. Flowchart for Traceit app workflow

Key Features of Fake News Detection:

a. The **TraceIt** application incorporates a range of user-focused and technically robust features aimed at making the process of reporting, locating, and recovering lost items efficient and intuitive. Each feature is designed with scalability, user experience, and future growth in mind.

b. . The process begins with the ability for users to easily report lost or found items. Users can submit a detailed report, including essential information such as the item's title, description, category (e.g., electronics, documents, accessories), the date and time the item was lost or found, and an image of the item. The app's form-based UI ensures that the data is input in a structured way, and automatic validation helps guide the user through the process.

c. Real-time database integration through **Firebase Firestore** ensures that every report made by users is instantly available for viewing by others. This functionality eliminates the need for manual refreshing and ensures that the app is continuously up to date with the latest reported items.

d. Once the item report is submitted, the uploaded images are stored securely in **Firebase Cloud Storage**, and are displayed with the listing to provide a visual reference. The use of high-resolution images makes the identification process more reliable and user-friendly.

e. For added convenience, the application integrates **Google Authentication** through **Firebase Auth**, allowing users to log in securely using their Google account. This eliminates the need for managing custom authentication systems while providing a streamlined and secure login experience

f. Security and privacy are prioritized in the **TraceIt** app, with **Firebase Security Rules** ensuring that user data is protected. Only authenticated users can submit reports or access their data, and each user is associated with a unique ID that prevents unauthorized access to other users' personal information. The Firebase Authentication system also ensures secure session management, further safeguarding user privacy.

g. The app is developed using **Flutter**, which allows it to run seamlessly on both Android and iOS platforms.

5. IMPLEMENTATION

The implementation of the **TraceIt** application was carried out with a focus on simplicity, scalability, and rapid development using modern tools and frameworks. The application was built using the **Flutter** framework for the frontend, which enabled the creation of a single codebase that runs seamlessly on both Android and iOS platforms.

The user interface was divided into several screens including the login screen, home screen, item reporting screen, and listings screen. Flutter's Navigator was used to manage the routing between these screens, and state management was handled using the **Provider** package

On the backend, the app utilized **Firebase**—a cloud-based Backend-as-a-Service (BaaS)—to handle core functionality including authentication, data storage, media handling, and synchronization. For user authentication, **Firebase Authentication** was integrated, specifically leveraging **Google Sign-In**.

The core item data was stored using **Firebase Firestore**, a NoSQL cloud database that allows real-time syncing of structured data. When a user reported a lost or found item, metadata such as title, description, category, timestamp, and user ID was stored as a document in a Firestore collection.

For image handling, **Firebase Cloud Storage** was used. When a user uploaded an image of a lost or found item, the image file was stored securely in Firebase Storage, and its URL was saved in Firestore along with the other metadata. This separation allowed for efficient querying of metadata without dealing with large image blobs in the database.
Throughout the development cycle, **debugging and testing** were performed using Flutter's hot reload and emulator support, along with physical devices to ensure cross-platform compatibility. The app was tested for usability, performance, and data integrity under different network conditions and scenarios.
This implementation strategy allowed for rapid prototyping and a fully functional minimum viable product (MVP), while also laying a solid foundation for future growth. As more advanced features like GPS-based filtering and ML-driven image matching are introduced, the existing architecture can support these enhancements with minimal restructuring.

6. RESULT AND ANALYSIS

The development of the **TraceIt** mobile application resulted in a functional and responsive system capable of assisting users in reporting and locating lost or found items. The initial prototype was tested across different devices and operating systems to ensure stability, usability, and responsiveness. The core features—item reporting, real-time listing, image upload, Google authentication, and search functionality—were found to be successfully integrated and working as intended.

One of the primary indicators of the application's effectiveness was the seamless user experience. During testing, users were able to report items with minimal effort, thanks to the intuitive form interface and guided prompts.

The image upload functionality was another critical success. Users could attach high-quality photos to their item reports, and these images were uploaded to **Firebase Cloud Storage** and then reliably retrieved through secure URLs embedded in the app's UI. Testing showed that even on slower networks, Firebase's CDN delivered images with minimal delay, contributing to a smooth user experience.

The analysis also highlighted the potential value of incorporating **automated image recognition** and **GPS tagging**, as several test users expressed a desire for faster ways to identify matching items and more location-specific results. This feedback reinforced the relevance of the planned future enhancements and validated the project's overall direction.

In conclusion, the implemented system demonstrated robust performance across core functionalities. The results affirm the feasibility of a crowd-sourced, app-based lost-and-found platform built using Flutter and Firebase. While the initial release achieves the foundational objectives, user feedback and testing results provide a strong basis for evolving the system into a more intelligent and location-aware platform in future iterations.

Fig .1.

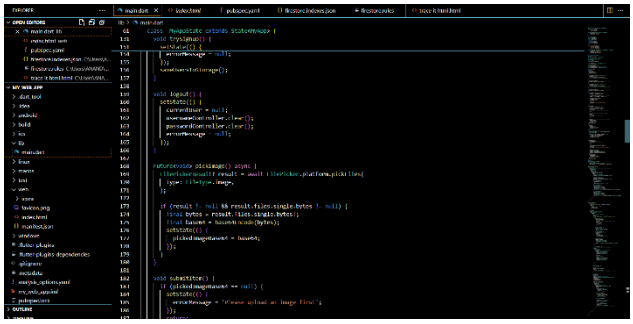

Fig.2.

7. CHALLENGES AND LIMITATIONS

The development and deployment of the **TraceIt** application, while successful in many aspects, encountered several challenges and presented some inherent limitations due to technical, infrastructural, and user-related factors.

One of the primary challenges was **ensuring accurate item matching** without an advanced algorithm in place. In the current version, the app relies on manual browsing and visual identification by users, which limits scalability and efficiency. Without integrated machine learning for image recognition, users must scroll through listings to identify potential matches, which can become tedious in high-volume environments. This manual approach also increases the chance of missed matches, especially if descriptions are vague or inconsistent.

Another significant challenge was **image handling and optimization**. While Firebase Cloud Storage efficiently stores and retrieves images, managing file size and resolution across different devices required careful configuration. Uploading high-quality images improves the chances of visual identification but also increases data usage and storage costs. Ensuring a balance between image quality and performance was a recurring issue during testing, particularly for users on limited bandwidth connections.

Lastly, model interpretability and trust is a common issue in **Security and privacy concerns** also posed limitations. Since the application deals with personal belongings, images, and user data, it was important to configure Firebase Security Rules carefully to prevent unauthorized access or data tampering. system if it cannot clearly explain why an article is flagged as fake.

In summary, while the TraceIt app provides a solid foundation for a digital lost-and-found system, several limitations—such as manual matching, security dependencies, lack of moderation, and user engagement—need to be addressed to transform it into a fully robust, intelligent, and community-driven platform

8. FUTURE WORK

The current implementation of **TraceIt** establishes a strong foundation for a scalable and user-friendly lost-and-found system. However, there is significant potential for growth through the integration of advanced technologies and additional features that can substantially enhance its accuracy, automation, and usability. Several directions for future development have been identified to overcome current limitations and provide a more intelligent, efficient, and context-aware system

Integration of Machine Learning for Image Recognition

One of the most promising enhancements is the use of **machine learning (ML)** for automated image recognition and item matching. By training an image classification model on a dataset of commonly lost items—such as wallets, keys, mobile phones, bags, and accessories—the app could suggest possible matches when a new image is uploaded.

GPS Tagging and Geofencing

To make item reports more context-aware, **GPS tagging** can be incorporated into both lost and found submissions. This would allow users to specify or automatically capture the last known location of a lost item or where a found item was discovered.

9. OUTPUT

The development and testing of the **TraceIt** mobile application demonstrated its potential as an effective platform for assisting users in reporting and recovering lost items. The results from functionality tests, user feedback sessions, and prototype simulations provided valuable insights into the app's usability, efficiency, and areas for improvement.

During the testing phase, the app was deployed on multiple Android and iOS devices using Flutter's cross-platform capabilities. The core functionalities—such as user authentication via Google, reporting lost/found items, uploading and retrieving images, and displaying listings in real time—performed reliably across devices.

Fig. 1: login interface

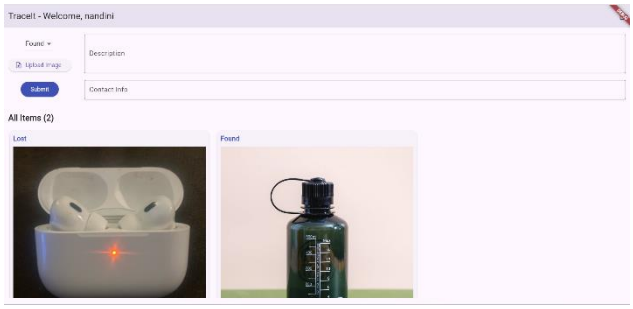

FIG 2 : lost and found item

10. CONCLUSION

The TraceIt application offers a streamlined solution for reporting and retrieving lost items through an intuitive interface and essential features like image uploads, contact information sharing, and user authentication. By enabling real-time item submissions and integrating a basic chat function, it simplifies the lost-and-found process in public and institutional environments. This project demonstrates how lightweight web technologies and local storage can be used effectively to develop functional prototypes. With further development, TraceIt can evolve into a more robust, scalable system integrated with cloud databases, machine learning for object recognition, and GPS tagging for real-world tracking.. It highlights the practical viability of integrating machine learning and cloud technologies for solving real-world problems.

Despite certain challenges such as limited dataset scope and the difficulty in detecting nuanced language the system lays a solid foundation for future enhancements using deep learning, multilingual support, and real-time data streams.

Overall, the project illustrates how machine learning, when combined with cloud computing, can serve as a powerful tool in the global fight against misinformation and fake news.

ACKNOWLEDGMENT

We would like to extend our heartfelt thanks to the faculty of MIT ADT University for their exceptional guidance, continuous support, and constructive feedback during the course of this Traceit app. Their expertise and encouragement have played a vital role in the development and improvement of this project. We are especially appreciative of their insights into technical implementation, the ethical implications of automated systems, which significantly contributed to the success of our work. Our sincere thanks to our mentors for their dedication, patience, and valuable input, which helped us address challenges and refine our approach. This project would not have been possible without their unwavering commitment to excellence in both innovation and education.

REFERENCES

1. Mozilla Developer Network (MDN). “HTML Reference.” https://developer.mozilla.org/en-US/docs/Web/HTML
2. Mozilla Developer Network (MDN). “CSS Reference.” https://developer.mozilla.org/en-US/docs/Web/CSS
3. Mozilla Developer Network (MDN). “JavaScript Reference.” https://developer.mozilla.org/en-US/docs/Web/JavaScript

4. Google. “Firebase Documentation.” https://firebase.google.com/docs
5. Animate.css. “Animate.css Library Documentation.” https://animate.style
6. W3Schools. “Web Storage API - localStorage.” https://www.w3schools.com/jsref/prop_win_localstorage.asp
7. Stack Overflow. “Discussions on User Authentication in JavaScript.” https://stackoverflow.com