

Lost Item Retrieval System

A Vasavi Sujatha, Siri Gouru, Spoorthy Yallamati, Sreeya Kotha

¹Assistant Professor, Department of Information Technology, Bhoj Reddy Engineering College for Women.

^{2,3,4}B.tech students, Department of Information Technology, Bhoj Reddy Engineering College for Women.

Sirigouru888@gmail.com

ABSTRACT

The Lost Item Retrieval System is a web-based platform designed to streamline the process of locating, reporting, and recovering lost and found items in environments such as colleges, offices, and libraries. Traditional manual tracking methods often lead to inefficiencies, errors, and data loss. This system replaces those with a centralized, real-time digital solution that enables users to report lost or found items, search through an integrated database, receive real-time notifications, and validate claims through a secure verification module. Key components include a smart matching algorithm, admin dashboard, and user-friendly interfaces that enhance usability and reliability. Built using technologies like HTML, CSS, Next.js, and Firebase, the system ensures scalability, security, and fast performance. Future developments aim to incorporate IoT, AI-based image recognition, mobile apps, and blockchain to further improve tracking accuracy and data integrity. This project highlights the potential of modern technology in solving common challenges. Keywords: Intrusion Detection System (IDS), denial-of-service (DoS).

1. INTRODUCTION

The Lost Item Retrieval System is a digital platform designed to streamline the process of reporting, tracking, and recovering lost items in institutions like colleges, offices, and libraries. It addresses the inefficiencies of traditional manual tracking by offering a centralized, real-time solution. Users can register lost or found items with detailed descriptions and search the database using filters like name, category, or location. The system enhances item recovery rates by employing smart matching algorithms. It also features real-time notifications and verification mechanisms to ensure that items are returned to their rightful owners. By reducing the time and effort required to find misplaced belongings, the platform significantly improves user convenience. The inclusion of an admin dashboard provides oversight and control for better system management. Designed with usability and scalability in mind, it caters to a wide range of users while maintaining security. This project demonstrates how technology can effectively solve everyday problems. Overall, the system brings efficiency, reliability, and user satisfaction to item management processes.

Existing system:

The existing system for managing lost and found items is primarily manual, relying on physical

registers, spreadsheets, or written logs to record item-related information. This method is inefficient and prone to various issues. Searching for lost items becomes a time-consuming task, often requiring individuals to check multiple locations or rely on staff assistance, which delays the retrieval process. Additionally, manual data entry increases the risk of human errors such as missing or incorrect records and duplicate entries. One of the major drawbacks is the absence of a centralized platform, making it difficult to search for items across different areas or departments. As a result, the likelihood of permanently losing items increases due to misplaced records and ineffective tracking.

Proposed System:

The proposed Lost Item Retrieval System offers a modern, efficient solution to the challenges faced by the existing manual processes. It provides a centralized, digital platform where users can quickly report lost or found items and search for them using an online database. The system features real-time tracking, allowing users to receive instant updates on the status and location of their items. A smart matching algorithm helps identify and suggest possible matches between lost and found reports, increasing the chances of successful recovery. Real-time notifications, an intuitive user interface, and an admin dashboard for managing reports and users further improve usability and efficiency. By automating and streamlining the entire process, the proposed system significantly reduces the time, stress, and errors involved in retrieving lost items.

2-RELATED WORK

In the Lost Item Retrieval System, this reference supports the future scope involving IoT integration. RFID tags or GPS trackers could be attached to valuable items (e.g., ID cards or electronics), enabling automated real-time location tracking. This technology would significantly enhance recovery speed and accuracy.[1] Ashton, K. (2009). That 'Internet of Things' Thing. RFID Journal. The matching module of your system can be upgraded using deep learning to match lost and found items based on textual descriptions or image inputs. For instance, a convolutional neural network (CNN) could identify items from uploaded images and match them accurately with reported entries.[2] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. Blockchain can be used in your system to maintain a secure and transparent ledger of all item transactions and ownership verifications. This can prevent fraudulent claims and ensure that ownership history is traceable.[3]

Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. This guide is relevant to the future expansion of your system as a mobile application. Mobile platforms can improve accessibility and engagement, allowing users to report or track items anytime and anywhere.[4] Android Developers. (n.d.). Build your first app. Security is a major concern in user data handling and ownership verification. Cryptographic methods like encryption, secure logins, and digital signatures can protect sensitive information and ensure only authorized users can access or claim items.[5] Stallings, W. (2017). Cryptography and Network Security: Principles and Practice. Pearson. You can apply these computer vision techniques to recognize and classify lost/found items from user-uploaded images. This would automate the identification process and increase the system's effectiveness.[6] Szeliski, R. (2010). Computer Vision: Algorithms and Applications. Springer.

3. REQUIREMENT ANALYSIS

Functional Requirements:

To build an effective Item Retrieval System, the following functional requirements must be implemented using appropriate technologies and methodologies:

- Item Registration
- Lost Item Reporting
- Found Item Reporting
- Search and Matching
- Real Time Notifications
- Verification System
- Admin Dashboard

3.2 Non-Functional Requirements:

3.2.1 Performance

- The system should provide fast response times for reporting, searching, and matching items.

3.2.2 Security

- All user data and item information must be protected from unauthorized access and misuse.

3.2.3 Scalability

- The system should be capable of handling an increasing number of users and items without performance degradation.

3.2.4 Usability

- The interface should be user-friendly and easy to navigate for users of all technical skill levels.

3.2.5 Reliability

- The system should operate consistently without crashes, errors, or data loss.

3.2.6 Maintainability

- The system should be easy to update, debug, and enhance without affecting existing functionality.

Software Requirements

Operating System : Windows
Frontend : HTML,
CSS,
Backend : Firebase
Web Development Framework : Next.js

3.3.2 Hardware Requirements:

Processor : Intel i5
RAM :
16GB
Storage : 250GB
SSD
System-type : 64-bit/32-bit
OS

4. DESIGN

4.1.1 System Architecture:

The system architecture of the **Lost Item Retrieval System** is built on a modern, modular design that ensures scalability, security, and maintainability. It follows a three-tier architecture consisting of the presentation layer, application layer, and data layer. The presentation layer serves as the front end of the application, developed using technologies like HTML, CSS, and the Next.js framework. This layer offers a user-friendly interface for reporting lost or found items, searching the database, receiving notifications, and managing user profiles. It is responsive and accessible from various devices, ensuring a smooth user experience.

The **application layer** acts as the intermediary between the user interface and the backend database. It handles core functionalities such as form validation, user authentication, session management, and smart item matching. This layer includes modules for real-time notifications, verification, and administrative controls. Technologies like Firebase Authentication and Next.js APIs are used to ensure secure and efficient processing of data. It also integrates logic for filtering, comparing, and suggesting item matches based on user input.

The **data layer** uses **Firebase Firestore**, a cloud-based NoSQL database, to store and retrieve item records, user details, and activity logs. Firebase also manages cloud storage for images and other uploaded files and provides real-time syncing capabilities to ensure that all users receive the latest data instantly. This enables real-time tracking of lost and found items and supports secure storage and retrieval with minimal latency.

The system also supports **modular connectivity**, where each feature—such as the admin dashboard, search and matching engine, or verification system—is designed as an independent yet integrated module. This ensures easy maintenance and future scalability,

such as integrating IoT, AI, or blockchain technologies.

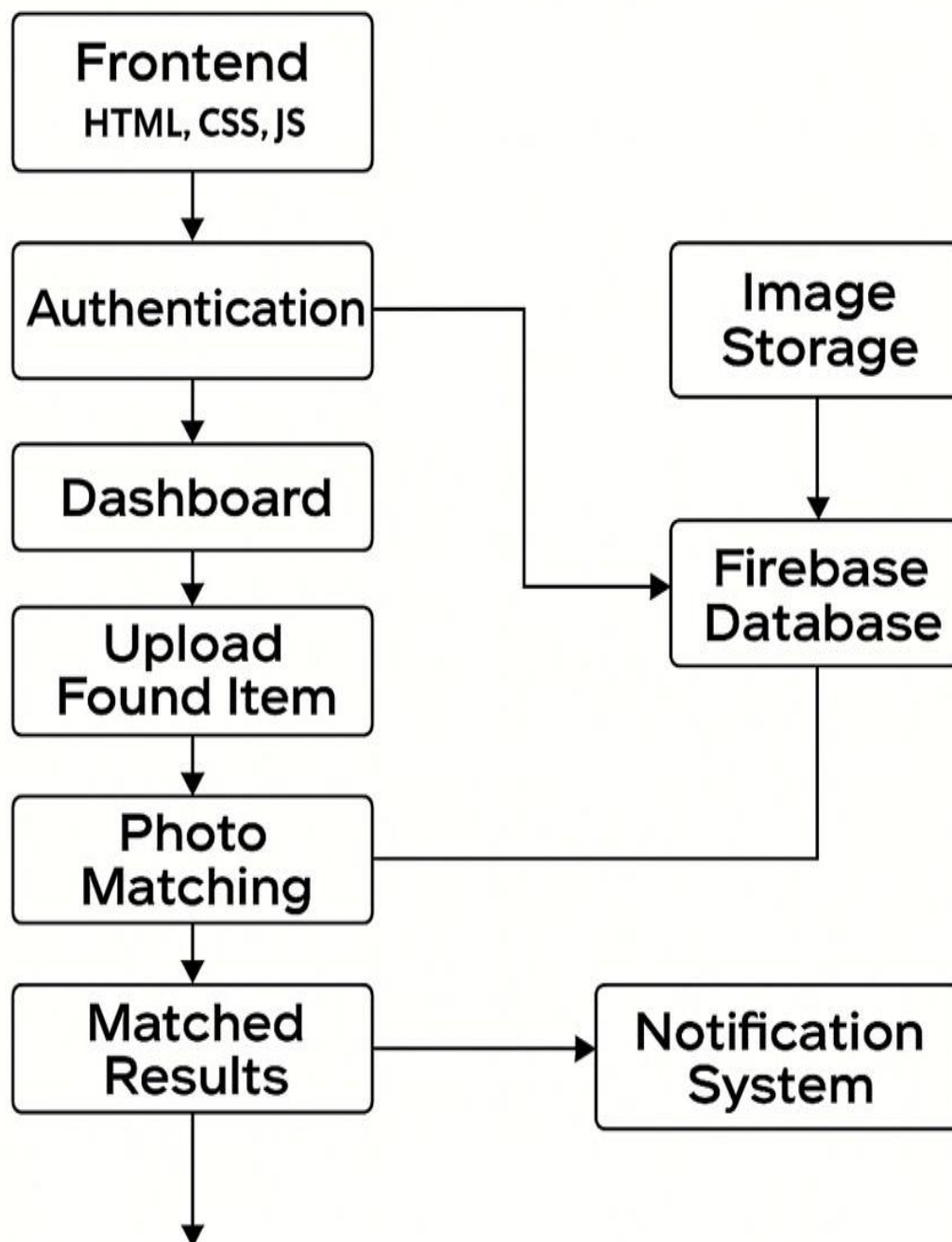


Fig. 4.1.1.1 System Architecture

4

1.2 Technical Architecture:

The **Lost Item Retrieval System** follows a modular, cloud-based, and event-driven architecture that leverages modern web technologies for enhanced performance and user interaction. The presentation layer (front end) is built using HTML and CSS for structure and styling, with Next.js, a powerful React-based framework, handling routing, rendering, and dynamic UI interactions. This framework ensures server-side rendering, fast page loads, and efficient client-side routing, making the platform accessible and responsive across devices. At the application layer, the system uses React components to handle user forms, inputs, and dynamic interfaces for features such as item registration, lost/found reporting, and admin dashboards. State management is handled efficiently through built-in hooks and external libraries. Real-time interaction is achieved using Firebase's Realtime Database and Firestore, which allow data to be stored and retrieved instantly across clients.

The backend services are powered entirely by Firebase, which acts as the core infrastructure. Firebase provides authentication services to securely manage user identities (e.g., email/password-based logins), cloud Firestore for structured data storage, and Cloud Functions for running backend logic, such as triggering notifications or verifying item claims. Firebase's serverless architecture reduces the overhead of managing servers and scales automatically based on the number of users and item reports.

In terms of data flow, user actions (like reporting a lost item) are captured by the front end, validated

using form validation libraries (such as Zod), and then sent to Firebase for storage. The system responds with real-time updates or notifications, which are pushed to users through Firebase Cloud Messaging (FCM) or email/SMS integrations. Each module—such as Search & Match or Admin Verification—interacts through a set of defined APIs and Firestore collections, ensuring data consistency and integrity.

For **security**, Firebase ensures encrypted communication via HTTPS, secured access through Firebase Authentication, and data protection using Firestore rules. Role-based access control is implemented to differentiate between users and admins. Sensitive user data like passwords are never stored in plain text and comply with privacy and protection standards.

The technical architecture supports a microservices-inspired structure, where each feature module (e.g., Lost Item Reporting, Real-Time Notifications) operates independently but integrates seamlessly. This design not only enhances maintainability and flexibility but also allows the system to grow with future extensions like mobile app integration, AI matching, or IoT-based tracking without major rework.

The Overall, the technical architecture is designed to offer high performance, scalability, security, and usability, aligning with the system's goal of providing a fast, user-friendly, and trustworthy platform for item retrieval in institutions and public spaces.

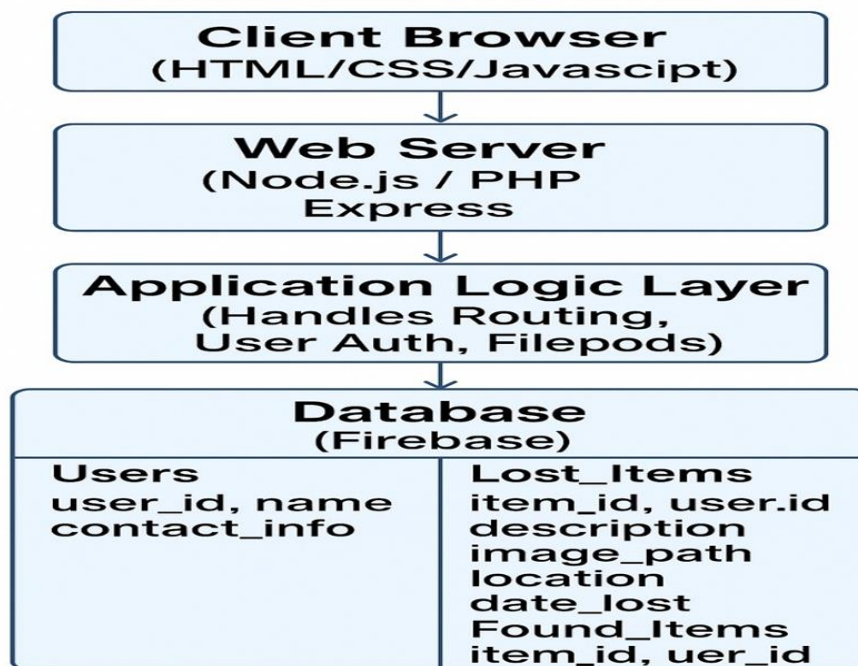


Fig. 4.1.2.1 Technical Architecture

5. IMPLEMENTATION

5.1 Libraries

5.1.1 Core Web Frameworks

Next.js & React

Next.js provides the application shell, enabling hybrid static + server-side rendering, file-based routing, and built-in optimisation for performance and SEO. Under the hood it is powered by React, which supplies the declarative component model that drives

every interactive screen in the app.

5.1.2 UI & Styling

Tailwind CSS

A utility-first CSS framework that lets developers compose modern, responsive layouts without leaving their HTML/JSX. It keeps style definitions close to the markup and eliminates bulky custom stylesheets.

shadcn/ui

A curated collection of accessible React components styled with Tailwind. It jump-starts development with ready-made building blocks—Buttons, Cards, Dialogs—while still letting teams customise the look via Tailwind tokens.

5.1.3 Form Handling & Validation

react-hook-form

Lightweight form state manager that stores field values in uncontrolled components, reducing re-renders and improving performance on large forms such as “Lost Item Report” or “Found Item Report”.

Zod

A TypeScript-friendly schema validator. Zod schemas declare field rules (length, pattern,

enum, cross-field equality) and plug directly into react-hook-form’s resolver, delivering instant front-end validation and reusable back-end guard clauses.

5.1.4. Cloud Back-End Services

Firebase acts as the serverless back-end:

Auth secures sign-up / login with email–password or federated providers.

Cloud Firestore stores structured data (items, users, claims) with sub-second real-time sync.

Cloud Functions run event-triggered logic—e.g., auto-matching lost/found posts or sending verification emails.

Cloud Messaging (FCM) pushes real-time notifications to web or mobile clients.

5.1.5. Animation & UX Polish

Framer Motion

Adds fluid, spring-based animations for page transitions, hover effects, or list re ordering—important for keeping the interface lively while users browse items or receive match alerts.

5.1.6. Date & Time Utilities

date-fns

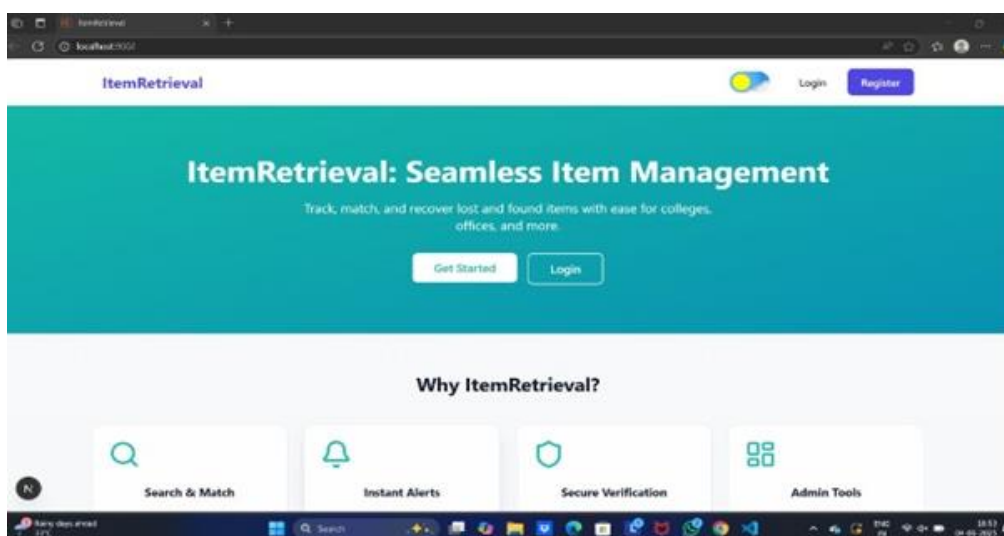
A lightweight alternative to Moment.js for formatting, parsing, and comparing dates. Used to stamp lost/found reports, display human-readable “2 hours ago” labels, and enforce date filters in search.

5.1.7. Icons & Graphics

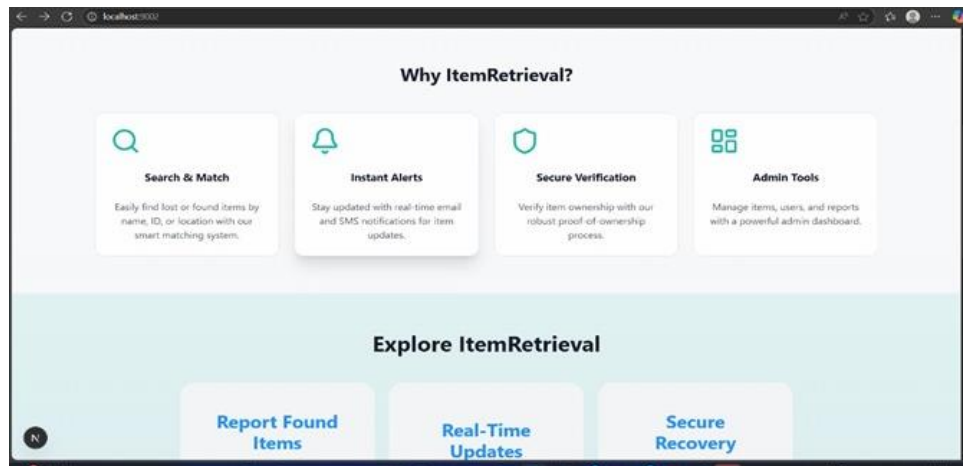
lucide-react

Open-source line-icons delivered as React components. Icons such as Calendar, LoaderCircle, and UserPlus provide visual cues without heavy asset payloads.

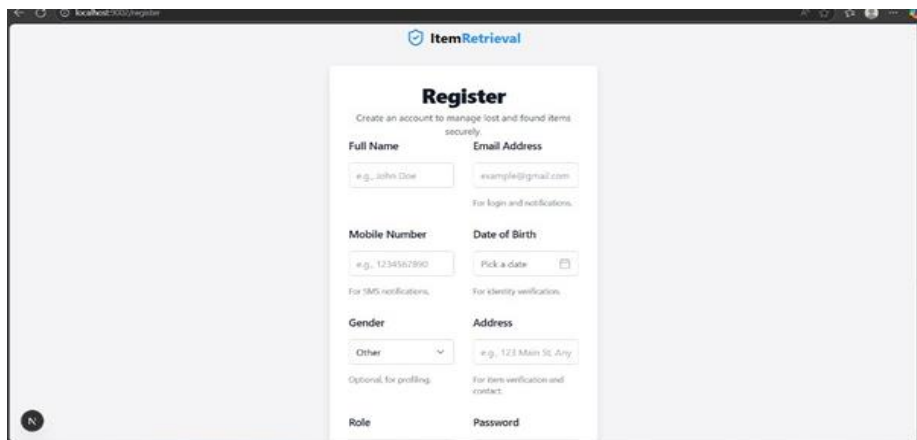
6. SCREENSHOTS



Screenshot 6.1 Home Page

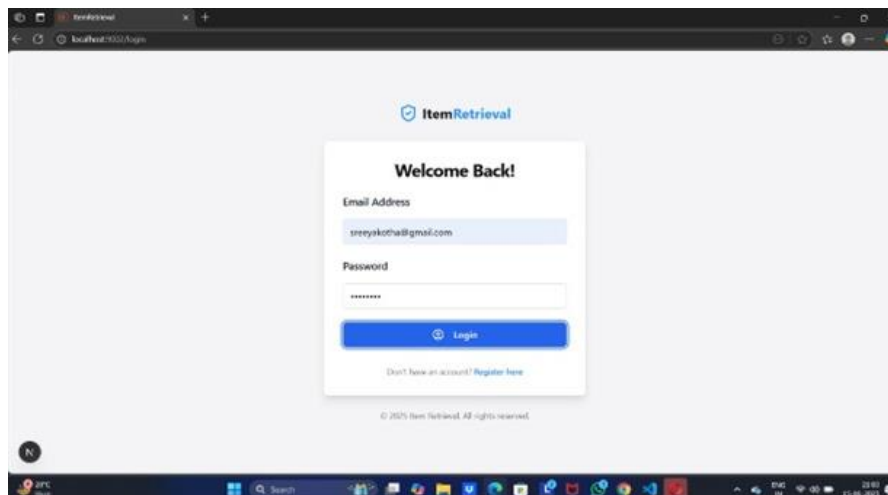


Screenshot 6.2 Data Insights



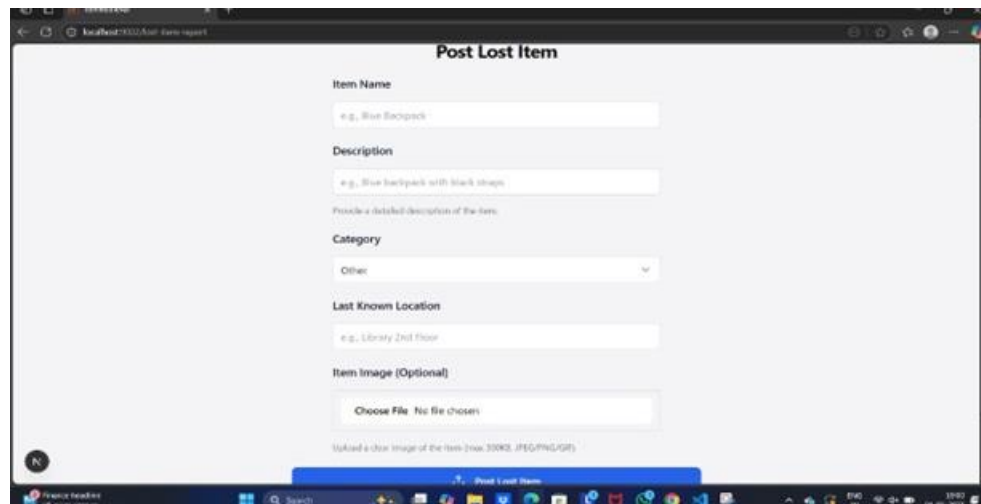
The screenshot shows a 'Register' form for the 'Item Retrieval' application. The form includes fields for 'Full Name' (e.g., John Doe), 'Email Address' (e.g., example@gmail.com), 'Mobile Number' (e.g., 1234567890), 'Date of Birth' (Pick a date), 'Gender' (Other), 'Address' (e.g., 123 Main St, Any), 'Role', and 'Password'. It also includes a 'Forgot Password' link and a 'Register' button.

Screenshot 6.3 Evaluation Page



The screenshot displays a 'Welcome Back!' login form for the 'Item Retrieval' application. The form includes fields for 'Email Address' (sreyakotha@gmail.com) and 'Password' (masked with dots). It features a 'Login' button and a link to 'Register here' for users who don't have an account. The footer indicates '© 2025 Item Retrieval. All rights reserved.'

Screenshot 6.4 Input page



Post Lost Item

Item Name
e.g., Blue Backpack

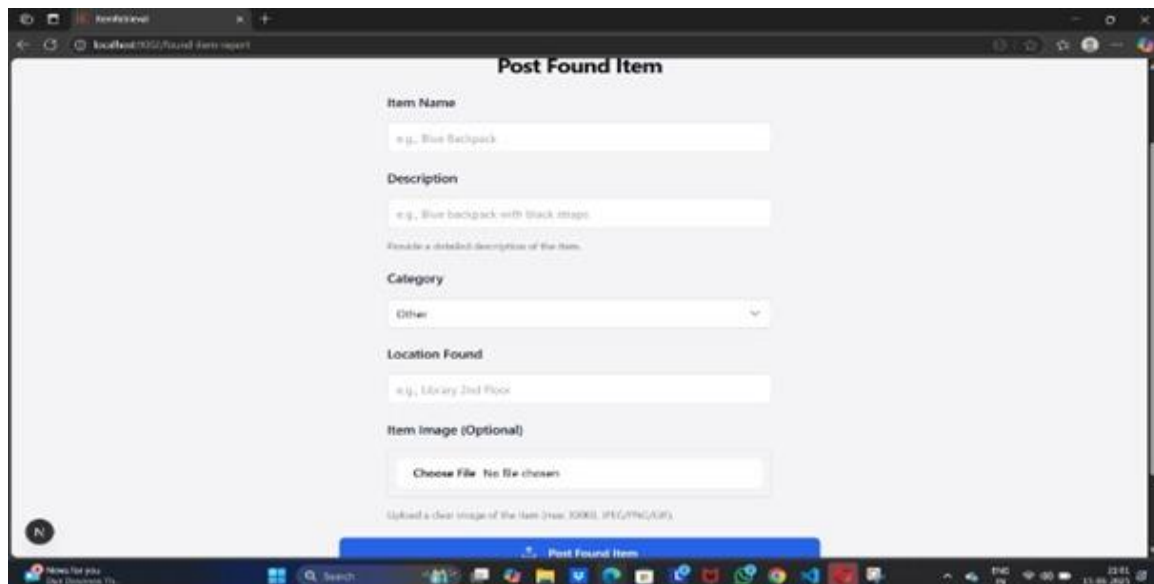
Description
e.g., Blue backpack with black straps
Provide a detailed description of the item.

Category
Other

Last Known Location
e.g., Library 2nd Floor

Item Image (Optional)
Choose File No file chosen
Upload a clear image of the item (max. 300KB, JPEG/PNG/GIF)

Screenshot 6.5 Final Output



Post Found Item

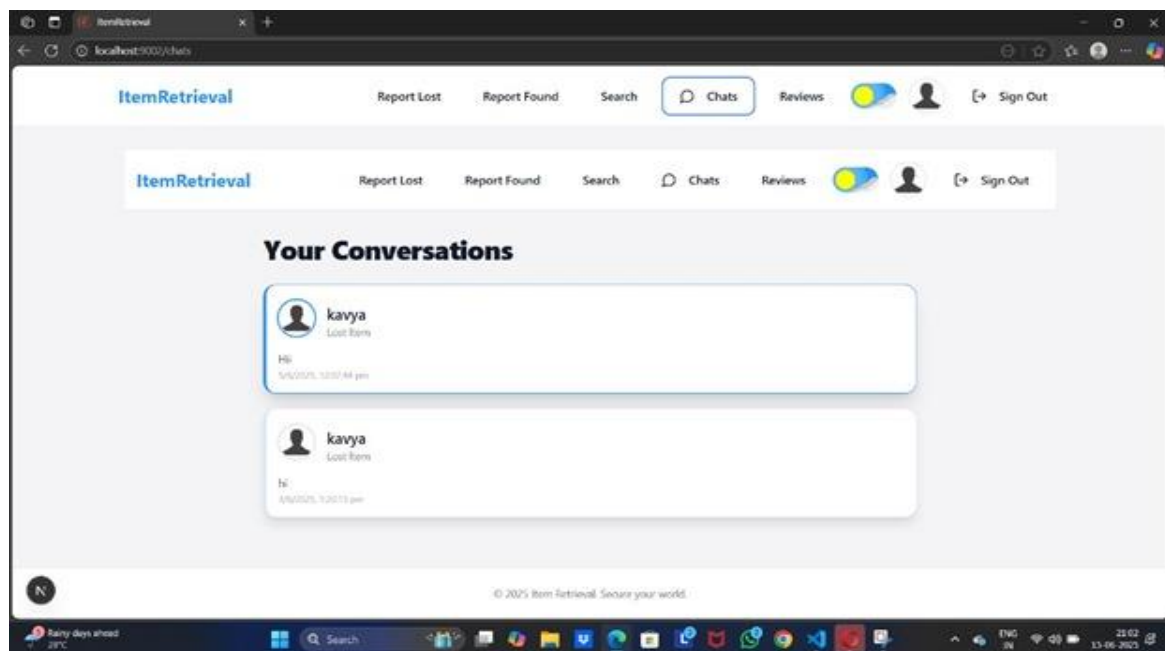
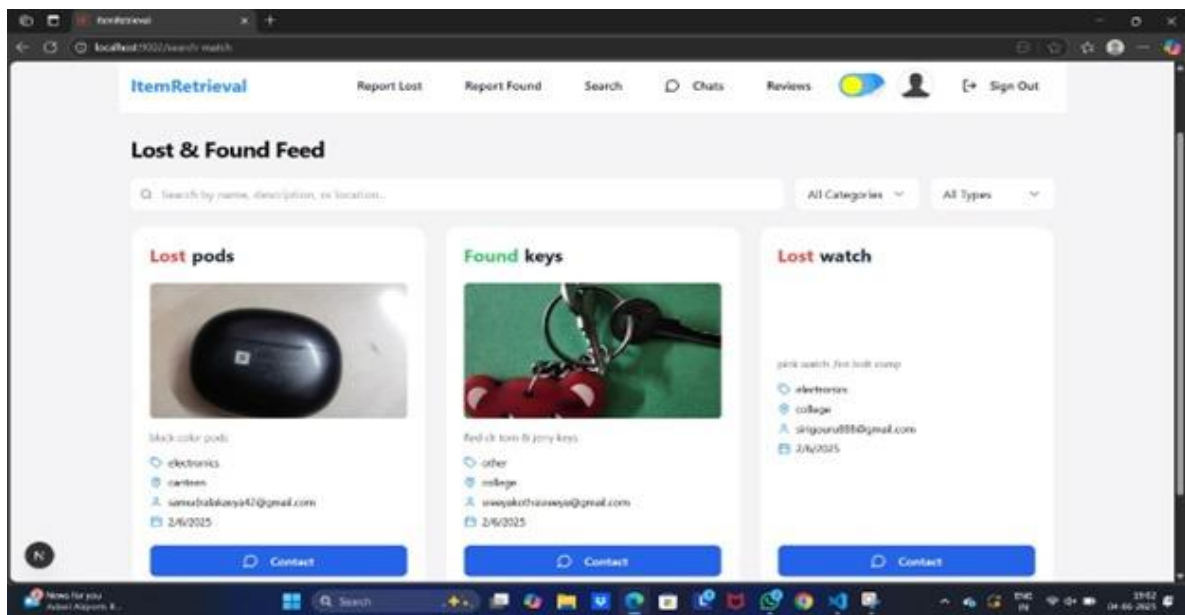
Item Name
e.g., Blue Backpack

Description
e.g., Blue backpack with black straps
Provide a detailed description of the item.

Category
Other

Location Found
e.g., Library 2nd Floor

Item Image (Optional)
Choose File No file chosen
Upload a clear image of the item (max. 300KB, JPEG/PNG/GIF)



7. CONCLUSION

The Lost Item Retrieval System offers a streamlined and efficient platform for reporting, searching, and managing lost and found items. By integrating modules such as Lost and Found Reporting, Search and Match, Verification, and an Admin Dashboard, the system improves the chances of item recovery and ensures secure and accurate communication between users.

With user-friendly interfaces, real-time notifications, and a secure login system, the platform enhances user experience and trust. This system is scalable, reliable, and plays a valuable role in reducing the time and stress involved in recovering lost items.

It demonstrates how technology can be used effectively to solve everyday problems with convenience and efficiency.

REFERENCES

- [1] Ashton, K. (2009). *That 'Internet of Things' Thing*. RFID Journal
- [2] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press
- [3] Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*.
- [4] Android Developers. (n.d.). *Build your first app*.
- [5] Stallings, W. (2017). *Cryptography and Network Security: Principles and Practice*. Pearson.
- [6] Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer.
<https://szeliski.org/Book/>