Lost n` Found

A Project Report
Submitted in partial fulfillment of the
Requirements for the award of the Degree of
BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)
By

Jatin Dnyaneshwar Gangare (9469)

Under the esteemed guidance of

Prof. Omkar Sherkhane

DEPARTMENT OF INFORMATION TECHNOLOGY

MAHATMA EDUCATION SOCIETY'S

PILLAI COLLEGE OF ARTS, COMMERCE & SCIENCE (AUTONOMOUS)

(*Affiliated to University of Mumbai)*

NEW PANVEL - 410206 , MAHARASHTRA

YEAR 2024-2025

MAHATMA EDUCATION SOCIETY'S

. Pillai College Of Arts Commerce & Science (Autonomous),

New Panvel

(Affiliated to University of Mumbai)

CERTIFICATE

This is to certify that the project is entitled **"Lost n` Found"**, is Bonafide work of **Mr. Jatin Dnyaneshwar Gangare** bearing Seat No: **9469,** submitted in partial fulfillment for the completion of the B.Sc. degree in Information Technology at the University of Mumbai.

Internal Guide **External Examiner**

College seal

Date: **Co-Ordinator**

ACKNOWLEDGEMENT

I, **Mr. Jatin Dnyaneshwar Gangare** student of **Pillai College Of Arts, Commerce & Science (Autonomous), New Panvel** would like to express my sincere gratitude towards our college's Information Technology Department.

I would like to thank Mrs. Deepika Sharma (Vice Principal) for granting me the opportunity to build a project. Last but not least I thank our guide Prof. Omkar Sherkhane for his constant support during this project. The project would have not been completed without the dedication, creativity and the enthusiasm my family provided me.

Yours faithfully,

JATIN DNYANESHWAR GANGARE
(Final Year Information Technology)

DECLARATION

I hereby declare that the project entitled, **"Lost n` Found"** done at **Pillai College Of Arts, Commerce & Science (Autonomous), New Panvel**, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project is done in partial fulfillment of the requirements for the award of degree of **BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)** to be submitted as final semester project as part of our curriculum.

Name and Signature of the Student

Mr. JATIN DNYANESHWAR GANGARE

Table Of Contents

CHAPTER 1
INTRODUCTION

1.1 Background

In rural areas like Kundevahal, managing lost and found items has traditionally been a challenging task. The process often involves manual methods such as word-of-mouth communication, announcements during village meetings, or notices displayed on bulletin boards. These methods are not only time-consuming but also prone to inefficiencies, miscommunication, and delays. Furthermore, there is no centralized or structured approach to track and verify lost items, resulting in many items being left unclaimed or returned to the wrong owners. This lack of an organized system has created a need for a reliable solution to address the issue.

The "Lost n Found" app was conceptualized to resolve this problem by leveraging the growing availability of smartphones and internet connectivity, even in rural areas like Kundevahal. The app introduces a digital platform that streamlines the process of managing lost and found items within the village. It incorporates a structured workflow where individuals who find lost items can submit them to an authenticator—a person appointed by the village representatives to oversee the process. This approach not only organizes the handling of lost items but also fosters a sense of community responsibility and cooperation.

The app distinguishes between two types of users: general users and authenticators. General users can report their lost items by filling out a detailed form that includes information such as the item’s description, original or look-alike image, Aadhar card details, and contact number. Authenticators play a critical role in verifying the details and managing the return of lost items to their rightful owners. With the use of Firebase Realtime Database, all records, including lost items, accepted requests, handover details, and final statuses, are securely stored and updated in real time, ensuring transparency and accuracy.

By introducing the "Lost n Found" app, the project aims to modernize and digitize the lost and found process in Kundevahal. This innovative solution reduces manual effort, builds trust through transparency, and enhances the overall efficiency of reuniting lost items with their rightful owners.

1.2 Objectives

1. Streamline the Process of Managing Lost and Found Items:

   To create a structured and efficient digital platform that simplifies the management of lost and found items in the village of Kundevahal, eliminating the inefficiencies and delays associated with traditional manual methods.

2. Facilitate User-Friendly Interaction:

   To develop an app that offers an intuitive and user-friendly interface, allowing users to easily report lost items, upload relevant details, and track the status of their items in real time.

3. Ensure Reliable Verification and Transparency:

   To establish a secure and transparent verification process managed by authenticators, ensuring that all lost items are returned to their rightful owners after thorough identity verification.

4. Enhance Community Engagement and Trust:

   To foster a sense of collaboration and trust among the villagers by introducing a centralized system that encourages responsible handling of lost items and reduces the chances of disputes or misunderstandings.

5. Leverage Technology for Rural Development:

   To demonstrate the practical use of modern technology, such as Firebase and Android development, in addressing everyday challenges faced by rural communities, bridging the gap between traditional practices and digital solutions.

1.3 Purpose

The purpose of the **"Lost n Found"** app is to provide a streamlined and efficient solution for managing lost and found items within the village of Kundevahal. The app aims to address the challenges associated with traditional methods, such as lack of organization, inefficiencies, and delays, by offering a centralized platform that enables the community to easily track and recover lost items. This initiative seeks to modernize the process while fostering trust and accountability among villagers.

This app is designed to simplify the experience for both users and authenticators. Users can report lost items by submitting essential details, such as item descriptions, images, and personal identification information, through an intuitive interface. Authenticators, appointed by village representatives, can then verify, approve, and manage the recovery process with transparency. The purpose of this structured workflow is to ensure that items are returned to their rightful owners with minimal hassle and maximum reliability.

The project also aims to strengthen community engagement by encouraging villagers to actively participate in the process of managing lost items. By involving trusted representatives as authenticators and providing real-time updates to users, the app builds a sense of collaboration and accountability within the village. This enhances mutual trust and ensures a fair and systematic approach to resolving lost and found cases.

Ultimately, this app serves as a practical example of how technology can be utilized to address everyday issues faced by rural communities. By bridging the gap between traditional practices and modern solutions, this app highlights the transformative potential of digital tools in improving quality of life and fostering community development.

1.4 Scope

Losing something can be frustrating, but finding it shouldn't be. Designed for the people of Kundevahal, this platform makes it easy to report, track, and recover lost belongings within the community. It operates with two key roles: general users, who report lost items, and authenticators, who verify and manage the recovery process. With a smooth and secure system in place, reuniting lost items with their owners becomes a hassle-free experience.

For users, the app provides features such as submitting details of lost items, including images, descriptions, and identification proofs, along with real-time updates on the status of their requests. Users can also delete their lost item report if they recover the item independently. The app ensures that users can track the progress of their request and receive notifications about handover dates and locations when their lost item is found.

For authenticators, the app includes tools to verify the ownership of lost items and manage their return. Authenticators can approve user requests, specify handover details such as date and time, and complete the final handover process by capturing a live photo of the user with the item and re-verifying their identity. All data, including lost item reports, accepted requests, and handover records, is securely stored in Firebase Realtime Database, ensuring accuracy and transparency.

The scope of the project extends beyond the immediate resolution of lost and found cases. It demonstrates how technology can address community-specific challenges, fostering trust, collaboration, and accountability. While currently focused on Kundevahal, the app’s framework can be adapted for other communities or regions with similar needs, showcasing its potential for broader application.

1.5 Applicability

This app is specifically designed to meet the needs of the village of Kundevahal, addressing the everyday challenge of managing lost and found items. Its primary application lies in creating a centralized and efficient platform that benefits both the residents who lose items and those who find them. By introducing a structured process with clear roles for users and authenticators, the app ensures smooth communication and accountability, making it highly suitable for small communities.

The app is versatile and can be adapted to other similar environments beyond Kundevahal. For instance, it can be applied in schools, offices, residential complexes, or public events where lost and found issues are common. Its ability to categorize and track items, verify ownership, and maintain records makes it an ideal solution for locations that require an organized and transparent system to manage lost items.

In addition to its practical use in tracking and recovering items, the app promotes collaboration and trust within the community. By involving village representatives as authenticators and providing real-time updates to users, the system fosters a sense of accountability and collective responsibility among its users. This makes it an effective tool for building stronger community ties while addressing a shared challenge.

Furthermore, the app's implementation showcases the potential for leveraging technology in rural areas. Its reliance on widely available smartphones and internet access highlights how modern tools can be adapted to solve specific community problems. This scalability and adaptability make this app a valuable model for addressing similar issues in other rural and semi-urban settings.

CHAPTER 2
SYSTEM PLANNING

2.1 Survey of Technologies

Firebase Platform

Firebase was selected as the backend for the **"Lost n Found"** app because of its robust features and seamless integration with Android. Its Realtime Database was a key choice for maintaining synchronized data for lost items, accepted requests, and handover records, all accessible in real time. Additionally, Firebase Cloud Storage enabled secure and scalable storage for essential images like item photos, Aadhaar card scans, and live verification photos. Firebase Authentication ensured secure and straightforward email-based login and registration for both users and authenticators, streamlining account management. Compared to alternatives like AWS or self-hosted solutions, Firebase stood out for its simplicity, cost-efficiency, and comprehensive developer tools, making it ideal for this project.

Android Studio

Android Studio served as the core development environment for building this app, offering a complete suite of tools for Android application development. Its visual layout editor facilitated the creation of a user-friendly interface, including intuitive forms for item submission and request tracking. The platform's support for Java, the chosen programming language for this project, ensured robust application development with access to extensive libraries and community resources. Android Studio’s built-in emulator enabled comprehensive testing across various device configurations, ensuring compatibility and performance. Its integration capabilities with Firebase further strengthened its role as the ideal choice for the app’s development.

Push Notifications

Push notifications were integrated into the app using Firebase Cloud Messaging (FCM) to provide real-time updates to users and authenticators. This feature was essential for keeping users informed about key events, such as the acceptance of their lost item requests, and notifying authenticators of newly submitted forms.

Programming Language

Java was chosen as the primary programming language for the app due to its reliability, widespread use in Android development, and strong community support. While Kotlin was considered for its modern features, Java was preferred for its stability and the developers' familiarity with the language. Java’s compatibility with Firebase SDKs and its rich ecosystem of libraries made it an ideal choice for building the app, ensuring a robust and maintainable codebase.

2.2 Fact-Finding Techniques

Fact-finding techniques were employed during the planning and development phases of the app to gather accurate and comprehensive information about the requirements, expectations, and potential challenges. These techniques helped ensure that the app effectively addressed the needs of the village of Kundevahal. The following methods were used:

1. Interviews

Direct interviews were conducted with key stakeholders, including village representatives, potential users, and appointed authenticators. These interviews provided insights into their expectations, challenges faced with tracking lost items, and specific features they wanted in the app. For example, the need for Aadhaar card verification and real-time notifications was identified during these discussions. Interviews helped bridge any communication gaps and ensured the app design aligned with the end-users' requirements.

2. Surveys and Questionnaires

Surveys and questionnaires were distributed to the residents of Kundevahal to collect feedback on their experiences with lost and found items. Questions focused on the frequency of lost items, the types of items commonly lost, and their preferences for recovering such items. The data gathered from this process highlighted the importance of features such as image uploads, detailed descriptions, and a streamlined authentication process.

3. Observation

Observation was carried out by analyzing existing methods used by villagers to handle lost and found items, such as announcements in community meetings or informal communication. By observing these traditional methods, gaps and inefficiencies were identified, such as the lack of systematic tracking and verification processes. These observations informed the app's features, including the use of a database for managing lost items and push notifications for updates.

2.3 Feasibility Study

The feasibility study evaluates the practicality of the app by assessing its technical, economic, operational, and social aspects. This ensures the app is viable, sustainable, and aligned with its objectives.

1. Technical Feasibility

The app leverages robust technologies such as Android Studio for development and Firebase for backend services, including Realtime Database, Cloud Storage, and Authentication. These tools ensure scalability, reliability, and ease of integration. The development team's expertise in Android and Firebase, along with the availability of necessary hardware like smartphones and internet connectivity, guarantees the technical viability of the app.

2. Economic Feasibility

The app is cost-effective due to the use of open-source tools and Firebase's free-tier services, which minimize development and maintenance expenses. For future scalability, any additional Firebase costs can be covered through local funding or contributions from village representatives. The app provides immense value by streamlining the process of tracking and recovering lost items, justifying its cost in terms of time and effort saved for the users.

3. Operational Feasibility

The app addresses a critical problem in Kundevahal by offering a systematic and efficient solution for lost item tracking. Its user-friendly interface ensures accessibility for users of all ages and technical backgrounds. Features like push notifications, Aadhaar-based authentication, and live photo verification enhance usability and security. The inclusion of village-appointed authenticators ensures smooth and trusted operations, reinforcing the app's practicality in daily use.

2.4 Stakeholders

The success of the **"Lost n Found"** app relies on the collaboration of various stakeholders, each of whom plays a crucial role in the app's development, implementation, and ongoing success.

1. Primary Stakeholders

- **Users**: The primary beneficiaries of the app, users are individuals from the village of Kundevahal who may lose items. They are responsible for submitting lost item reports, providing necessary documentation, and interacting with the app to track the recovery of their lost items.

- **Authenticators**: Appointed by village representatives, authenticators are responsible for verifying the lost item claims, accepting requests for items, and overseeing the handover process. They play a critical role in maintaining the app's integrity by ensuring only rightful owners receive their lost items.

- **App Developers**: As the sole developer of this project, all aspects of coding, design, and maintenance are handled independently. Ensuring the app functions smoothly and effectively, the development work focuses on meeting the needs of both users and authenticators, providing a seamless and reliable experience for the community.

2. Secondary Stakeholders

- **Village Representatives**: Local authorities or leaders who oversee the app's implementation and encourage its use in the community. They ensure the app's success by educating the village about its benefits and ensuring the authenticity of authenticators.

- **Firebase Service Providers**: Firebase offers backend services like real-time database, cloud storage, and authentication. Their platform enables the app's core functionality, and their role is essential for the app's data storage and real-time interactions.

- **Community Members**: The wider village population benefits from the app by ensuring a safer environment where lost items are more likely to be found and returned. Their participation in reporting lost items or helping others is vital for the app's success.

CHAPTER 3
REQUIREMENTS AND ANALYSIS

3.1 Problem Definition

In many communities, lost and found items often remain unclaimed due to the absence of an effective tracking system. Individuals who lose valuable items or documents struggle to retrieve them, while those who find items are left without a proper way to connect with the original owner. This problem is especially prevalent in rural areas, where there are no formal mechanisms to report or claim lost items.

Currently, the process of reporting and retrieving lost items is disorganized, relying on informal methods such as word of mouth or physical noticeboards. This can lead to confusion, miscommunication, and delays in returning the item to its rightful owner. Moreover, a lack of verification can result in disputes over ownership.

This situation calls for a digital solution that can streamline the process of reporting lost items, verifying claims, and facilitating the handover of found items. The goal is to create a reliable, efficient, and secure system for managing lost and found items, which can be easily accessed by individuals and verified by designated authorities within the community.

Key challenges include:

- Ensuring data security and privacy for both users and finders.
- Creating a straightforward, user-friendly interface for reporting and retrieving items.
- Implementing a secure method for verifying ownership of lost items.
- Providing a transparent process for the item handover, ensuring trust between the parties involved.

By addressing these issues, the solution will create a more efficient way for individuals to recover their lost items while ensuring transparency and reducing the risk of disputes.

3.2 Requirements Specification

3.2.1 Functional Requirements

The "Lost n Found" app allows users to register, log in, and report lost items by filling out a form that includes item details, images, and the owner's contact information. This data is stored in Firebase's Lost Items database. Authenticators can verify items, accept claims, and set pick-up times and locations. Upon item handover, the authenticator fills a Handover Form with the receiver's details and verifies their identity. Push notifications will update both users and authenticators about the status of items. Users can also delete lost items if they are found before the claim process.

3.2.2 Non-Functional Requirements

The app should be responsive (data retrieval within 3 seconds), user-friendly, and compatible with Android devices running Android 5.0 or later. It must handle real-time push notifications and be scalable for future growth. Security is crucial, and sensitive data such as Aadhar card images will be encrypted. The app should be available 24/7, with minimal downtime, and ensure compliance with data privacy regulations like GDPR.

3.2.3 System Requirements

The app requires Android devices with at least 2GB of RAM and an internet connection. It will be developed using Android Studio and use Firebase for authentication, real-time database storage, and push notifications. Firebase services must be available for proper functioning.

3.2.4 Constraints

The app is limited to Android devices and relies on an active internet connection for full functionality. It must comply with data privacy regulations, especially when handling sensitive information like Aadhar card images.

3.3 Planning and Scheduling

The "Lost n Found" project was planned to be completed over 22 weeks, with clear goals and deliverables for each phase. The first phase, lasting five weeks, involved thorough requirement gathering and analysis, followed by the initial design phase where the system architecture was defined, Firebase was set up for the backend, and UI mock-up’s were created. This phase laid the foundation for all future development.

The second phase, spanning five weeks, focused on the development of core features such as user authentication, handling of lost item forms, push notifications for both users and authenticators, and the integration of Firebase’s Realtime Database and Storage. This phase aimed at establishing the basic functionality and interaction between the mobile app and backend systems.

The third phase, lasting five weeks, was dedicated to improving the user interface and user experience (UI/UX) by incorporating feedback from initial testers. This phase focused on making the app easy to navigate, visually appealing, and responsive. It also included integration of the app's UI with the backend functionalities to ensure smooth operation.

In the fourth phase, lasting four weeks, the focus shifted to comprehensive testing and debugging. This involved identifying and fixing any bugs, testing performance under different conditions, and ensuring the app met all security standards, especially for sensitive data like Aadhaar numbers and images. Usability testing was also conducted to ensure that the app was intuitive for both users and authenticators.

Finally, in the last three weeks, the app was deployed to the Google Play Store. This phase also included post-deployment maintenance, monitoring the app’s performance, and providing ongoing support to users. The team worked on addressing any reported issues and providing timely updates based on user feedback.

Gantt Chart

Sr No.	Task	Month	June			July				August			September	
		Date	15	22	29	6	13	20	27	3	10	17	7	14
1	INTRODUCTION													
	Background	Planned												
		Actual												
	Objectives	Planned												
		Actual												
	Purpose	Planned												
		Actual												
	Scope	Planned												
		Actual												
	Applicability	Planned												
		Actual												
2	SYSTEM PLANNING													
	Survey of Techniques	Planned												
		Actual												
	Fact-Finding Techniques	Planned												
		Actual												
	Feasibility study	Planned												
		Actual												
	Stakeholders	Planned												
		Actual												
3	REQUIREMENTS AND ANALYSIS													
	Problem Definition	Planned												
		Actual												
	Requirement Specification	Planned												
		Actual												
	Planning & scheduling	Planned												
		Actual												
	Software & Hardware req.	Planned												
		Actual												
	Conceptual Model	Planned												
		Actual												
4	SYSTEM DESIGN													
	Basic Modules	Planned												
		Actual												
	Data Design	Planned												
		Actual												
	Procedural Design	Planned												
		Actual												
	User Interface & Design	Planned												
		Actual												
	Security Issues	Planned												
		Actual												
	Test Case Design	Planned												
		Actual												

Sr No.	Task	Month	November			December				January		
		Date	16	23	30	7	14	21	28	4	18	25
5	IMPLEMENTATION AND TESTING											
	Implementation and approaches	Planned										
		Actual										
	Coding Details and code Efficiency	Planned										
		Actual										
	Testing approach	Planned										
		Actual										
6	CONCLUSION AND FUTURE WORK											
	Conclusion and Future Scope	Planned										
		Actual										
7	REFERENCES											
	References	Planned										
		Actual										

3.4 Software and Hardware Requirements

Creating the "Lost n Found" application involved utilizing a blend of software tools and hardware components to achieve efficient functionality and seamless performance. Below is an overview of the key requirements:

- Software Requirements
    1. Operating System: Windows 10 or later to support development and testing activities.
    2. Android Studio: Integrated development environment used for building and designing the Android application.
    3. Firebase: Backend solution for managing authentication, real-time database operations, and cloud storage.
    4. Java and XML: Essential programming languages for app functionality and user interface development.
    5. Version Control: Git, to manage source code efficiently and facilitate collaboration.
    6. Testing Tools: Android Emulator, for testing app performance across various devices.
- Hardware Requirements
    1. Development Machine: HP 14 Laptop equipped with an Intel Core i5 processor, 8 GB RAM, and 256 GB SSD for optimal development performance.
    2. Mobile Device: Android device running version 7.0 or higher for live testing and validation.
    3. Internet Connection: Stable high-speed internet to enable Firebase integration and effective app testing.

3.5 Preliminary Product Description

This project aims to address the issue of managing and retrieving lost items efficiently within a community. The system is designed for use in a specific village, Kundevahal, and features two primary user roles: Users and Authenticators. Users can report lost items by filling out a detailed form that includes item descriptions, images, and contact details. Authenticators, appointed by the community, are responsible for verifying, listing, and managing these reported items.

The application provides a structured process where lost items are submitted, verified, listed, and matched with claims. Push notifications enhance communication between users and authenticators, ensuring timely updates. The system leverages Firebase for real-time database operations, cloud storage, and authentication, enabling seamless and secure data handling.

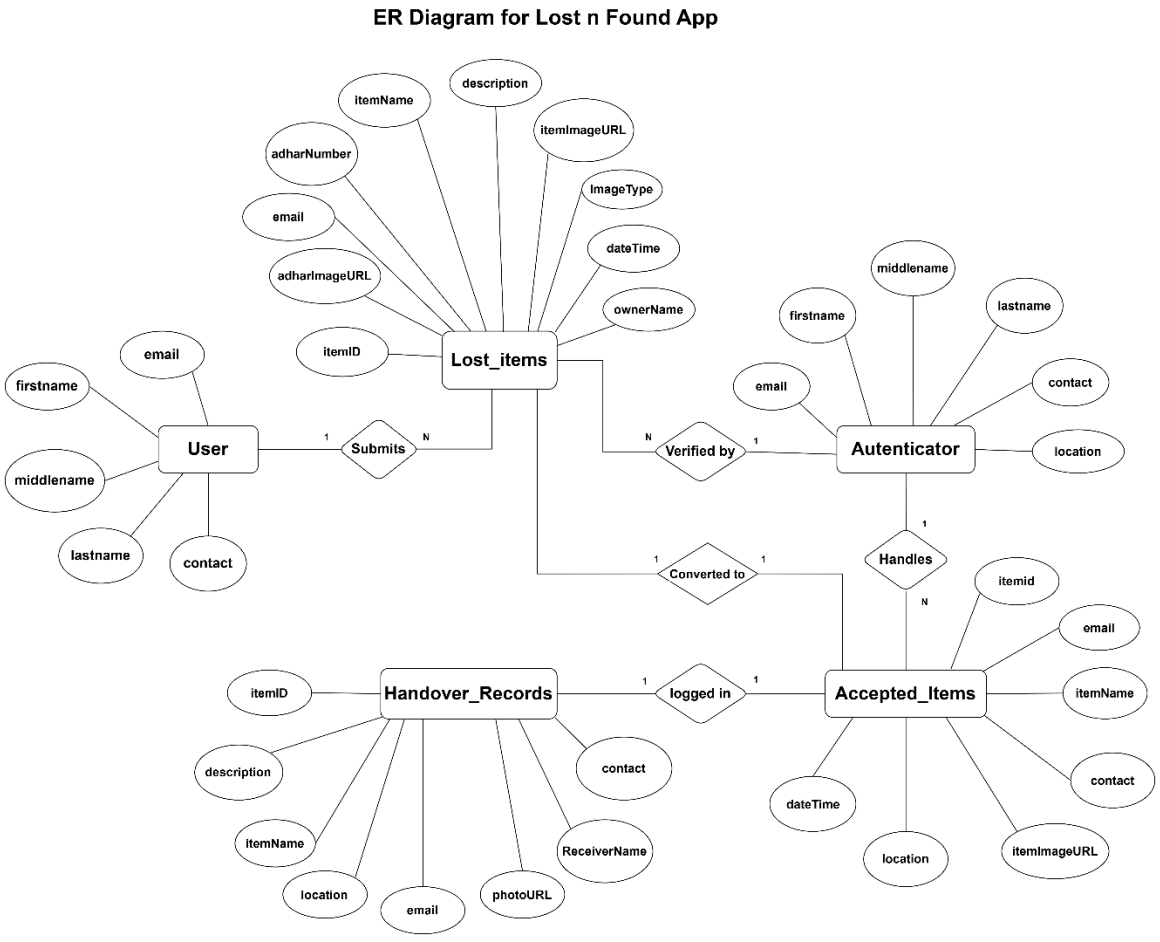
By digitizing the process of finding and returning lost items, the product aims to streamline operations, minimize delays, and foster a more organized approach to lost-and-found management in the community.

3.6 Conceptual Models

Conceptual Models are simplified, abstract representations of a system or process that highlight its key components, relationships, and interactions. They are used to visualize and communicate the structure and behavior of a system at a high level without delving into technical details.

3.6.1 ER Diagram

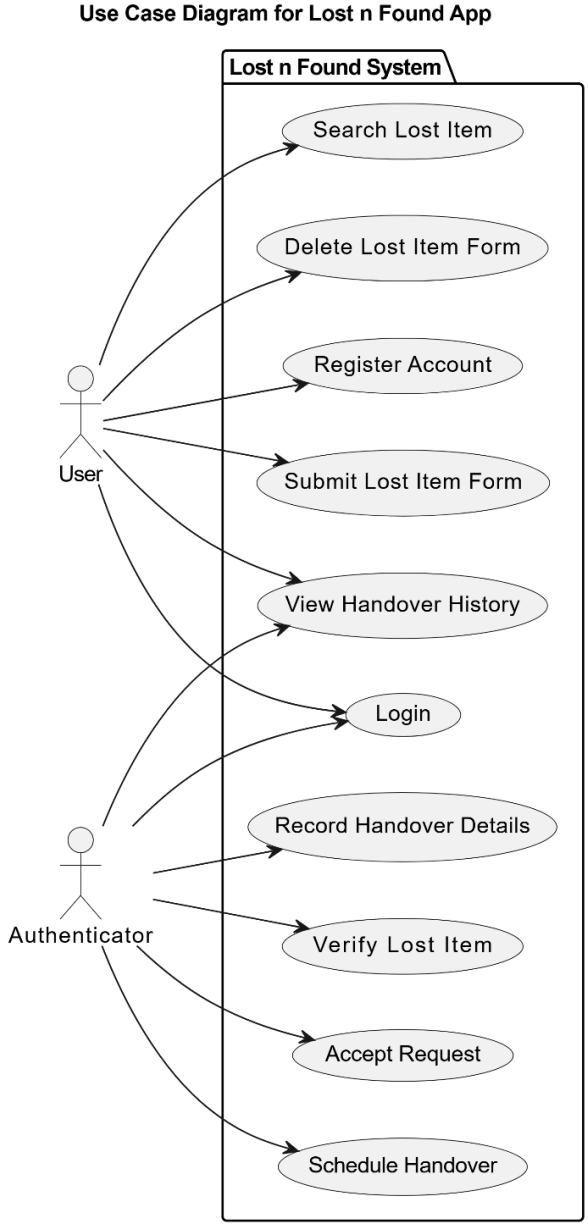
The ER diagram represents the core entities and relationships within the Lost n Found app. The User entity captures user details, such as email, name, and contact. Users can submit multiple lost item reports, represented by the Lost_Items entity, which includes details like item ID, description, and associated Aadhaar card information. These reports are verified by an Authenticator, who manages the Accepted_Items entity, detailing items ready for handover, including their location and verification status. The Handover_Records entity logs final item handovers, storing information about the recipient, item, and authentication details. The relationships ensure seamless tracking and verification throughout the process.



3.6.2 Use Case Diagram

The Use Case Diagram for the "Lost n Found" app illustrates the interactions between two primary actors: User and Authenticator within the system. Users can register an account, log in, submit a lost item form, search for lost items, delete a lost item form, and view handover history.

Authenticators, who manage lost item verification, can log in, verify lost items, accept user requests, schedule handovers, record handover details, and view handover history. This diagram highlights the essential functionalities of the system, ensuring a structured workflow for tracking and recovering lost items efficiently.



3.6.3 Class Diagram

The Class Diagram for the Lost n Found app outlines the key classes and their relationships within the system. It includes major components such as AuthenticatorMainActivity, HomeFragment, AddItemFragment, ListedItemFragment, LoginActivity, ProfileFragment, SignupActivity, UserMainActivity, HandoverFormFragment, and VerifyClaimFragment.

1. AuthenticatorMainActivity: This class serves as the main activity for the authenticator. It manages the lists of lost and accepted items, handles Firebase authentication (mAuth), and provides navigation methods like onCreateView(), loadItems(), and filter(). The authenticator can also redirect to login or add item fragments.

2. HandoverFormFragment: This fragment is used by the authenticator to confirm the handover of a lost item to the user. It includes methods to upload the handover image (uploadImageAndSaveRecord()), save the handover details to Firebase, and delete accepted items if needed.

3. HomeFragment: This is the primary fragment seen by users to view the list of items. It fetches and filters the lost items from Firebase (loadItems()), and offers navigation to other fragments like AddItemFragment and ProfileFragment.

4. AddItemFragment: Here, users can submit new lost items. The fragment allows users to fill in details such as item name, description, and location, take a photo, and save the item to Firebase using the method saveItemToDatabase(). The fragment also enables image capturing and item submission.

5. LoginActivity: This activity manages user login and redirects based on user type (Authenticator or User). It verifies credentials using Firebase authentication and checks network availability. It also allows users to initiate password recovery.

6. ProfileFragment: This fragment displays and allows editing of the user's profile. It includes methods for deleting user accounts and re-authenticating users. It also integrates a Google Map to show the user's location.

7. SignupActivity: In this activity, new users can register. It collects user details like name, contact, email, and password, and saves them to Firebase. It navigates to the login activity once the user signs up.

8. UserMainActivity: This is the main activity for the user. It includes methods for managing user sessions, listening for new accepted items, and navigating to other fragments like HomeFragment.

9. VerifyClaimFragment: This fragment is used by the authenticator to verify and accept a claim for a lost item. It checks the item details and allows the authenticator to set a time and date for the handover. It saves the acceptance details to Firebase.

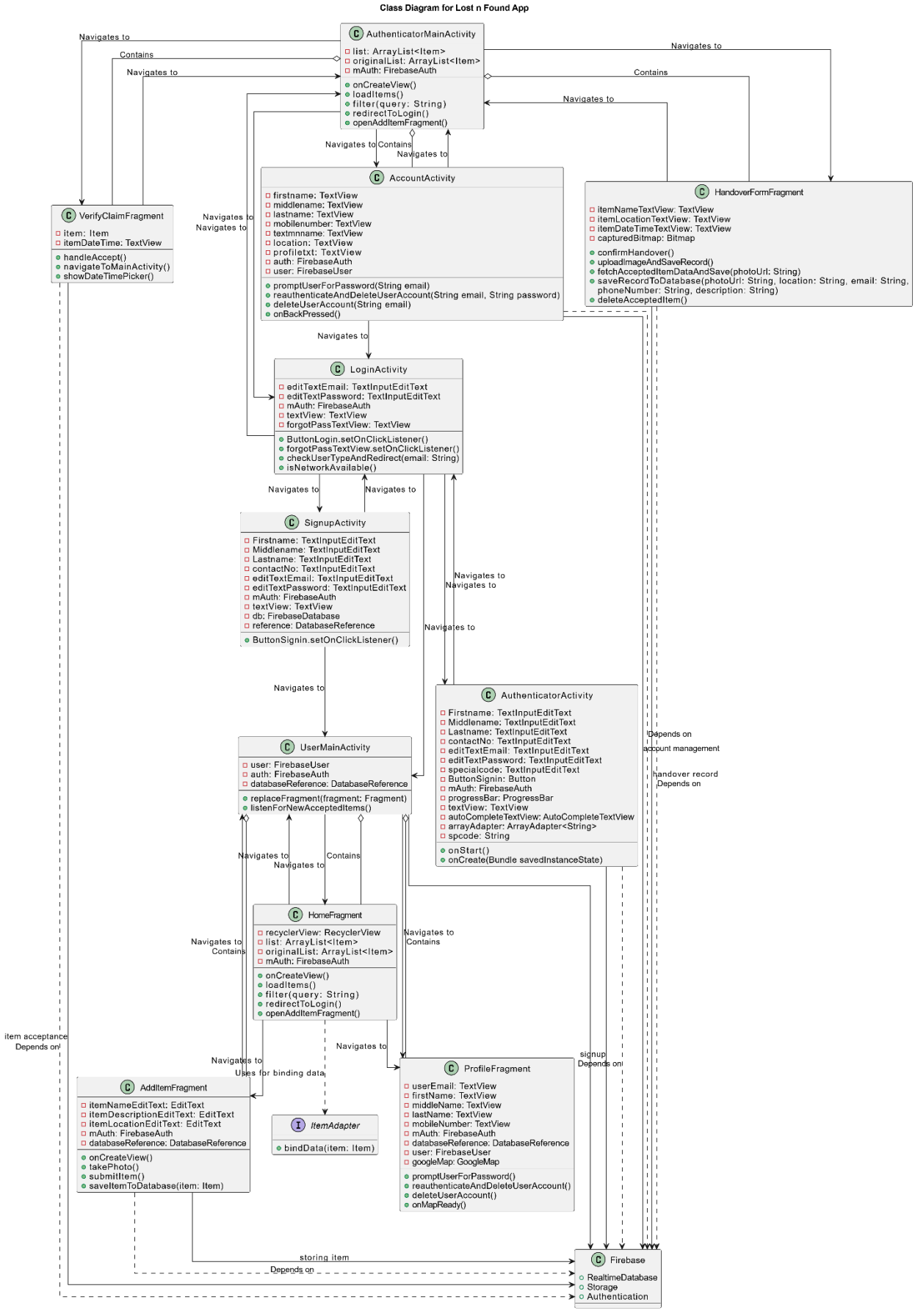
10. AccountActivity: Manages user account details and deletion, displaying fields like name, contact, and location. It includes methods for password authentication, reauthentication, and account deletion via Firebase, ensuring secure account management.

Relationships between Classes:

- The AuthenticatorMainActivity can navigate to multiple fragments, including HomeFragment, ProfileFragment, SignupActivity, and others.
- HomeFragment acts as a central hub, leading to other fragments like AddItemFragment, ProfileFragment, and ListedItemFragment.
- LoginActivity allows navigation to either the AuthenticatorMainActivity or SignupActivity, depending on the context.
- AddItemFragment and HandoverFormFragment both interact with Firebase to store or update item details.
- SignupActivity and LoginActivity handle user authentication and redirect between each other.
- UserMainActivity is responsible for managing user-related activities and navigation.

Class Interaction:

- The system heavily relies on Firebase for storing and retrieving data, particularly for lost items, accepted items, and handover records.
- Navigation between fragments is a key feature, allowing users and authenticators to transition seamlessly between activities like adding new items, verifying claims, or updating their profiles.


Class Diagram for Lost n Found App
AuthenticatorMainActivity
list: ArrayList<Item>
originalList: ArrayList<Item>
mAuth: FirebaseAuth
onCreateView()
loadItems()
filter(query: String)
redirectToLogin()
openAddItemFragment()
AccountActivity
firstname: TextView
middlename: TextView
lastname: TextView
mobilenumber: TextView
textmnname: TextView
location: TextView
profiletxt: TextView
auth: FirebaseAuth
user: FirebaseUser
promptUserForPassword(String email)
reauthenticateAndDeleteUserAccount(String email, String password)
deleteUserAccount(String email)
onBackPressed()
HandoverFormFragment
itemNameTextView: TextView
itemLocationTextView: TextView
itemDateTimeTextView: TextView
capturedBitmap: Bitmap
confirmHandover()
uploadImageAndSaveRecord()
fetchAcceptedItemDataAndSave(photoUrl: String)
saveRecordToDatabase(photoUrl: String, location: String, email: String, phoneNumber: String, description: String)
deleteAcceptedItem()
VerifyClaimFragment
item: Item
itemDateTime: TextView
handleAccept()
navigateToMainActivity()
showDateTimePicker()
LoginActivity
editTextEmail: TextInputEditText
editTextPassword: TextInputEditText
mAuth: FirebaseAuth
textView: TextView
forgotPassTextView: TextView
ButtonLogin.setOnClickListener()
forgotPassTextView.setOnClickListener()
checkUserTypeAndRedirect(email: String)
isNetworkAvailable()
SignupActivity
Firstname: TextInputEditText
Middlename: TextInputEditText
Lastname: TextInputEditText
contactNo: TextInputEditText
editTextEmail: TextInputEditText
editTextPassword: TextInputEditText
mAuth: FirebaseAuth
textView: TextView
db: FirebaseDatabase
reference: DatabaseReference
ButtonSignin.setOnClickListener()
AuthenticatorActivity
Firstname: TextInputEditText
Middlename: TextInputEditText
Lastname: TextInputEditText
contactNo: TextInputEditText
editTextEmail: TextInputEditText
editTextPassword: TextInputEditText
specialcode: TextInputEditText
ButtonSignin: Button
mAuth: FirebaseAuth
progressBar: ProgressBar
textView: TextView
autoCompleteTextView: AutoCompleteTextView
arrayAdapter: ArrayAdapter<String>
spcode: String
onStart()
onCreate(Bundle savedInstanceState)
UserMainActivity
user: FirebaseUser
auth: FirebaseAuth
databaseReference: DatabaseReference
replaceFragment(fragment: Fragment)
listenForNewAcceptedItems()
HomeFragment
recyclerView: RecyclerView
list: ArrayList<Item>
originalList: ArrayList<Item>
mAuth: FirebaseAuth
onCreateView()
loadItems()
filter(query: String)
redirectToLogin()
openAddItemFragment()
ProfileFragment
userEmail: TextView
firstName: TextView
middleName: TextView
lastName: TextView
mobileNumber: TextView
mAuth: FirebaseAuth
databaseReference: DatabaseReference
user: FirebaseUser
googleMap: GoogleMap
promptUserForPassword()
reauthenticateAndDeleteUserAccount()
deleteUserAccount()
onMapReady()
AddItemFragment
itemNameEditText: EditText
itemDescriptionEditText: EditText
itemLocationEditText: EditText
mAuth: FirebaseAuth
databaseReference: DatabaseReference
onCreateView()
takePhoto()
submitItem()
saveItemToDatabase(item: Item)
ItemAdapter
bindData(item: Item)
Firebase
RealtimeDatabase
Storage
Authentication
Navigates to
Contains
Depends on
account management
handover record
item acceptance
signup
storing item
Uses for binding data

3.6.4 Sequence Diagram

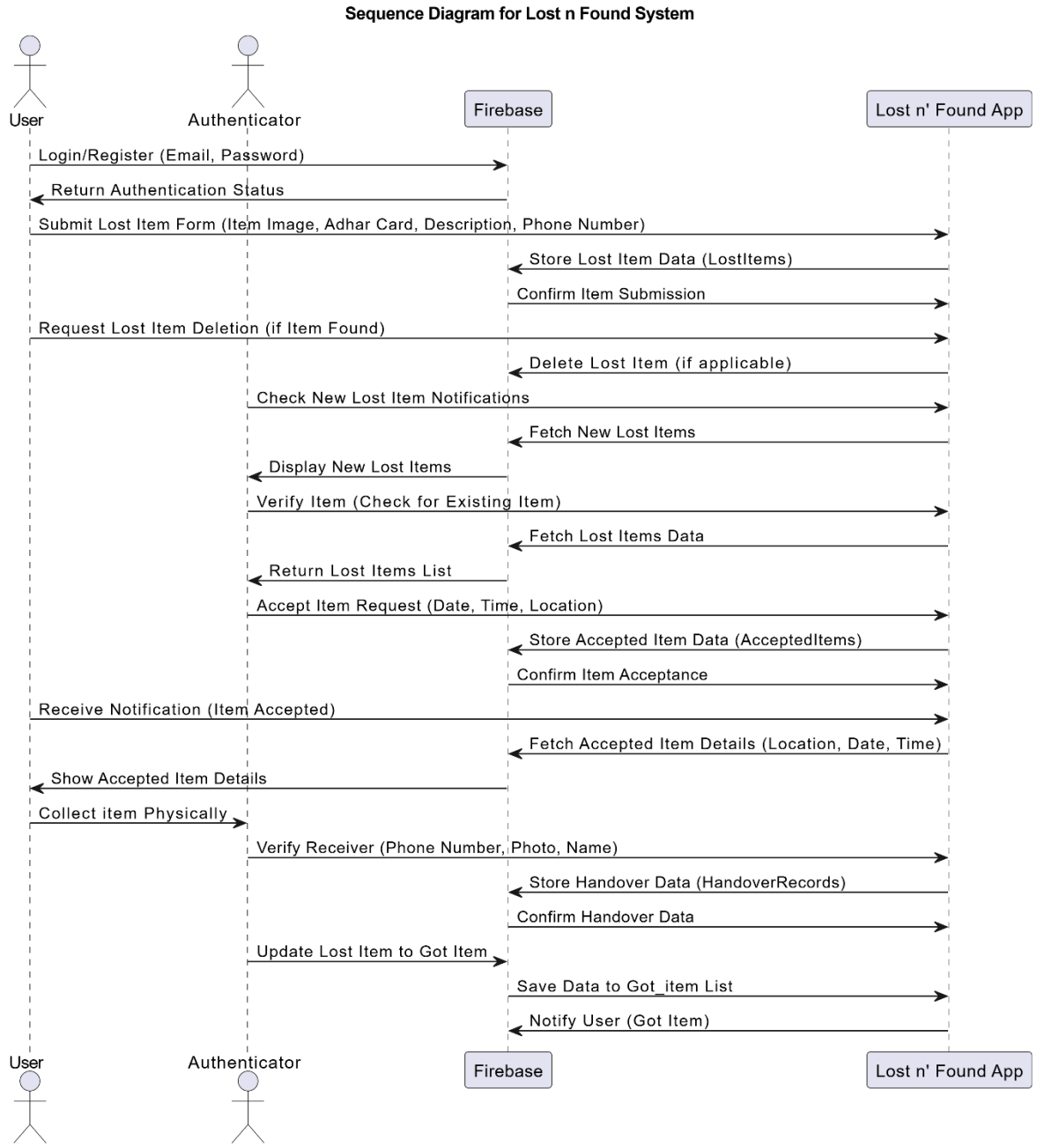
The benefits of Sequence diagram :

1. Clear Process Flow: It visually represents how users, authenticators, and Firebase interact, making it easier to understand the app's functionality.
2. Enhanced Collaboration: Facilitates communication between developers and stakeholders, ensuring everyone is on the same page.
3. Error Detection: Helps identify potential gaps or issues in the process, improving the design.
4. System Design: Aids in organizing backend and frontend workflows, ensuring efficient app operation.
5. Documentation: Provides a clear reference for future development and maintenance of the app.

The User begins by logging in or registering through the Lost n' Found App using their email and password. Once authenticated by Firebase, the user submits a lost item form, which includes details such as item images, Adhar card, description, and contact information. This data is stored in Firebase under the LostItems database. If the item is found, the user can request its deletion from the system, and Firebase updates the database accordingly.

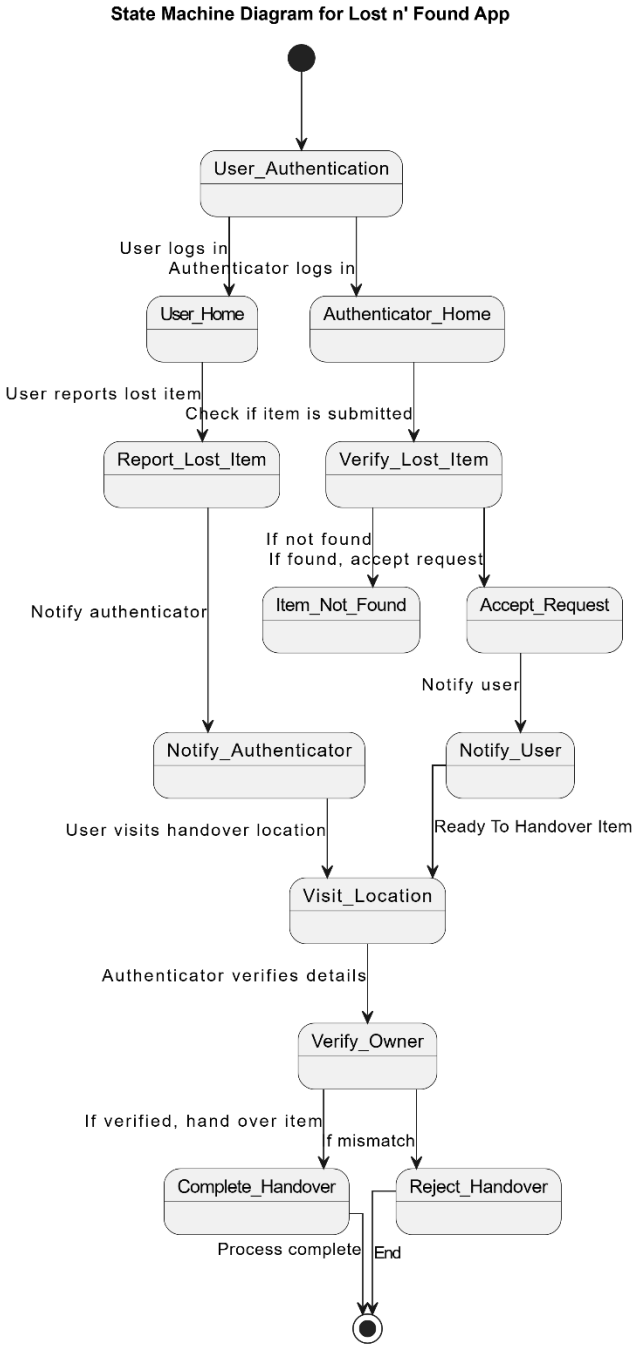
Meanwhile, the Authenticator, responsible for verifying lost items, checks the app for new lost item notifications, which are fetched from Firebase. The authenticator verifies whether the lost item has already been submitted by someone else. If the item is valid, the authenticator accepts the user's request by setting the handover date, time, and location, which is then stored in Firebase under the AcceptedItems list. The User receives a notification with the item acceptance details.

When the user arrives at the specified location to collect the item, the authenticator verifies the user's identity by checking their phone number, taking a photo, and confirming the name. This information is recorded in Firebase under HandoverRecords. After the item is handed over, the authenticator updates the item's status to "claimed" in the GotItems list in Firebase. Finally, the app sends a notification to both the user and the authenticator, confirming that the item has been successfully claimed, completing the process.


Sequence Diagram for Lost n Found System
User
Authenticator
Firebase
Lost n' Found App
Login/Register (Email, Password)
Return Authentication Status
Submit Lost Item Form (Item Image, Adhar Card, Description, Phone Number)
Store Lost Item Data (LostItems)
Confirm Item Submission
Request Lost Item Deletion (if Item Found)
Delete Lost Item (if applicable)
Check New Lost Item Notifications
Fetch New Lost Items
Display New Lost Items
Verify Item (Check for Existing Item)
Fetch Lost Items Data
Return Lost Items List
Accept Item Request (Date, Time, Location)
Store Accepted Item Data (AcceptedItems)
Confirm Item Acceptance
Receive Notification (Item Accepted)
Fetch Accepted Item Details (Location, Date, Time)
Show Accepted Item Details
Collect item Physically
Verify Receiver (Phone Number, Photo, Name)
Store Handover Data (HandoverRecords)
Confirm Handover Data
Update Lost Item to Got Item
Save Data to Got_item List
Notify User (Got Item)
User
Authenticator
Firebase
Lost n' Found App

3.6.5 State Machine Diagram

The State Machine Diagram of the Lost n' Found app outlines the process of reporting, verifying, and handing over lost items. Users report lost items, which are stored in Firebase and notified to the authenticator. The authenticator verifies the submission and either accepts or keeps it in the lost list. Upon acceptance, a handover date is set, and the user is notified. At collection, the authenticator verifies the user's Aadhaar and phone number, captures a live photo, and hands over the item if verified. The transaction is recorded, and the item is marked as retrieved. If verification fails, the handover is rejected.



3.6.6 Deployment Diagram

The Deployment Diagram for the Lost n' Found App illustrates the system's infrastructure, involving three main components: User Device, Authenticator Device, and Firebase Cloud Platform. The User and Authenticator Devices run the Android app, which interacts with Firebase services for authentication, database storage, image uploads, and push notifications. The Firebase Cloud Platform handles user authentication (Firebase Authentication), data storage (Firebase Realtime Database), image storage (Firebase Storage), and notifications (Firebase Cloud Messaging - FCM). Connections show how each component interacts, ensuring seamless data flow and communication between users, authenticators, and the cloud.

The system consists of three primary nodes:

1. User Device

The User Device runs the Lost n' Found App (Android), allowing users to submit and search for lost items.

The app ensures secure user authentication by communicating with Firebase Authentication to verify user credentials. It efficiently manages lost item records by interacting with Firebase Realtime Database for data storage and retrieval. To provide visual reference, lost item photos are uploaded to Firebase Storage. Additionally, the app keeps users informed through real-time notifications from Firebase Cloud Messaging (FCM), ensuring they receive timely updates on item status changes.

2. Authenticator Device

The Authenticator Device also runs the Lost n' Found App (Android) but with additional functionalities to verify and manage lost items. The authenticator interacts with Firebase services similarly to the user device but with added responsibilities.

The authentication process in the app ensures that only authorized users can access the system by verifying authenticator credentials through Firebase Authentication. For lost item verification, the app retrieves data from Firebase Realtime Database, allowing authenticators to review lost item reports and approve claims based on provided details. When an item is accepted, its images are uploaded to Firebase Storage, serving as verification proof for future reference. Additionally, the app keeps users informed by sending real-time notifications through Firebase Cloud Messaging (FCM), updating them on the status of item verification and handover processes.

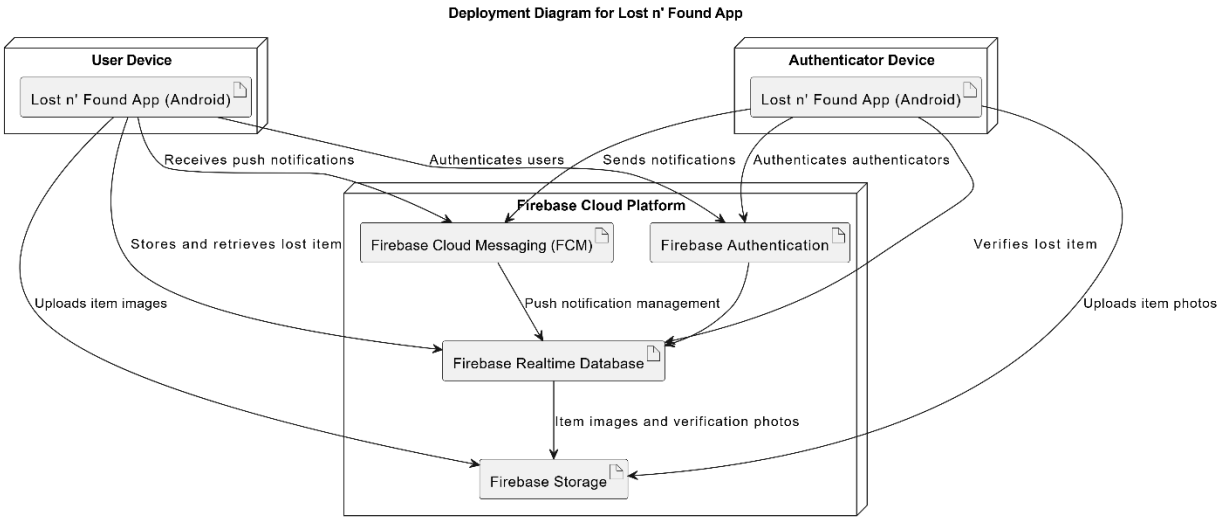
3. Firebase Cloud Platform

The backend of the system is built on Firebase, which consists of several components handling different functionalities.

Firebase Authentication securely manages login credentials for both users and authenticators, ensuring authorized access to the app. Firebase Realtime Database stores essential information, including user details, lost item records, accepted requests, and handover history. To handle media files efficiently, Firebase Storage is used for uploading and storing item images and verification photos. Additionally, Firebase Cloud Messaging (FCM) enables real-time push notifications, keeping users updated on the status of their lost items.

Interactions Between Components

- The User Device and Authenticator Device both connect to Firebase services for authentication, data storage, image uploads, and notifications.
- Firebase Authentication ensures only registered users and authenticators can access the app.
- Firebase Realtime Database synchronizes item-related data between users and authenticators.
- Firebase Storage keeps a record of item images and verification photos.
- Firebase Cloud Messaging (FCM) sends notifications for updates like accepted items, claim verification, and handover confirmations.



3.6.7 System Flowchart

The system ensures a smooth workflow, including user authentication, lost item submission, verification by authenticators, and the final handover process. Below is a detailed breakdown of each section in paragraph format.

1. User Opens the App

The process begins when the user launches the Lost n' Found App. The system first determines the type of user logging in—whether they are a regular User or an Authenticator. Based on this classification, the user is directed to the appropriate interface and features available to their role.

2. Authenticator Flow

If the user is an Authenticator, they log into their account and gain access to the list of lost items submitted by users. The authenticator reviews new lost item requests and verifies the details against the existing records in the system. If the item is found in the database, the authenticator proceeds to accept the request, assigns a specific handover date and time, and notifies the user that their lost item request has been approved. The details of this action are logged into the accepted_items database for tracking.

3. User Flow

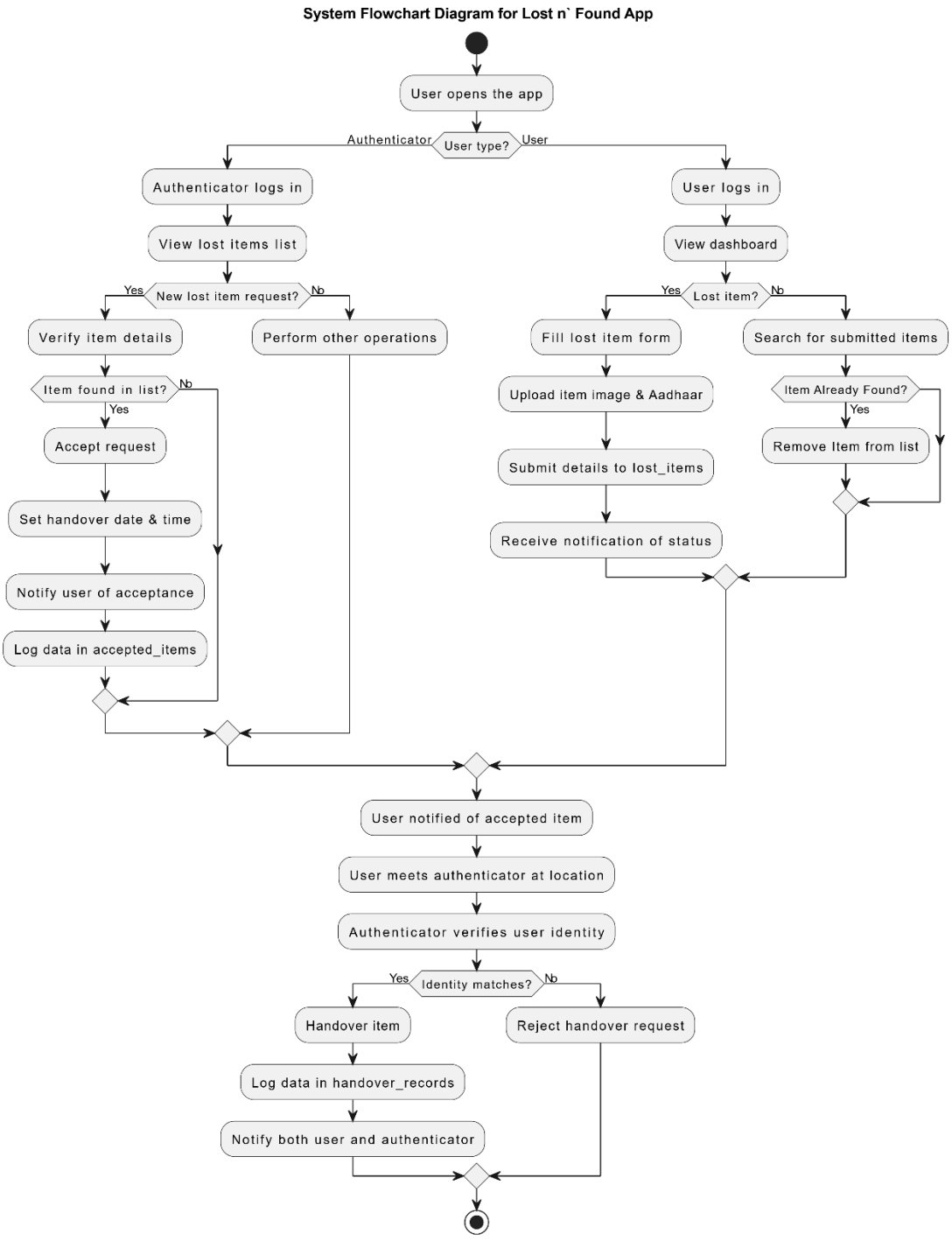
If a regular user logs in, they are directed to their dashboard, where they can report a lost item or search for previously submitted lost items. To report a lost item, they fill out a form with details like item description, location, photo, and Aadhaar ID for verification. The details are stored in the lost_items database, and the user receives a status notification. Users can also search for lost items, and if an item is found, they can remove it from the list to keep the database updated with active cases only.

4. Item Collection Process

Once an authenticator accepts a lost item request, the user is notified and must meet at the assigned location. The authenticator verifies the user's identity using Aadhaar ID or other identification. If verified, the item is handed over, and the details are recorded in the handover_records database, with both parties receiving confirmation. If not, the request is rejected, ensuring only rightful owners claim lost items.

5. End of Process

After the handover process is completed, the system records the transaction and ensures that all relevant details are stored in the database. The process then ends, marking the completion of the lost item retrieval journey. This flowchart ensures that the Lost n' Found App maintains a secure, reliable, and well-structured system for handling lost items, verifying claims, and ensuring proper tracking of lost and found items in the system.



CHAPTER 4
SYSTEM DESIGN

4.1 Basic Modules

The **Lost n' Found System** is designed to facilitate the process of reporting, verifying, and retrieving lost items. The system consists of multiple core modules, each serving a specific function to ensure seamless interaction between users, authenticators, and the system. Below are the fundamental modules of the system:

1. User Authentication Module

This module is responsible for user registration and authentication. It allows users to create an account, log in securely, and manage their credentials. Firebase Authentication is used to validate and store user credentials securely.

2. Lost Item Submission Module

Users can report lost items by providing necessary details such as item description, location, date, and an image. This information is stored in the Firebase Realtime Database under the Lost Items section.

3. Item Verification Module

Authenticators review submitted lost items to verify their legitimacy. They check item details and cross-verify with found items before accepting a claim request. Once verified, the item is moved to the Accepted Requests section in the database.

4. Notification Module

This module sends real-time notifications to users and authenticators about updates regarding lost item submissions, verification statuses, and claim approvals using Firebase Cloud Messaging (FCM).

5. Item Handover Module

Once an item is verified, the authenticator schedules a handover with the user. During the handover, the authenticator fills a verification form with user details and item photos, ensuring a proper record is maintained in the Handover Records section.

6. Database Management Module

The system uses Firebase Realtime Database to store and manage users, authenticators, lost items, accepted requests, handover records, and retrieved items. This ensures real-time updates and efficient data handling.

7. Search & Retrieval Module

Users can search for lost items based on keywords, categories, or location. If an item is found, the user can claim it, triggering the verification process by the authenticator.

4.2 Data Design, Data Integrity and Constraints

You're correct! Since Firebase Realtime Database is a NoSQL database, it does not explicitly enforce primary keys, foreign keys, or constraints like relational databases (SQL). Instead, Firebase uses a JSON-based hierarchical data structure where each entity is stored as a key-value pair. However, we can still maintain data integrity by structuring the database properly and implementing rules at the application level.

Revised Data Design for Firebase Realtime Database

Instead of SQL-style tables, Firebase Realtime Database organizes data into nodes (collections) and subnodes (documents). Below is how the entities will be structured:

1. Users Collection

Users/{userId}
Each user is stored under a unique userId, which acts as an implicit primary key.


```
{
  "Users": {
    "user123": {
      "contact": "9876543210",
      "email": "user@example.com",
      "firstname": "Jatin",
      "lastname": "Gangare",
      "middlename": "Dnyaneshwar"
    }
  }
}
```


Integrity Rules:

- userId (Firebase-generated UID) is unique and used for authentication.
- Email is stored for validation, but uniqueness is enforced via Firebase Authentication.

2. Authenticator Collection

Authenticator/{authenticatorId}
Similar to Users, each Authenticator has a unique ID (authenticatorId).


```
{
```

```
 "Authenticator": {
   "auth123": {
     "contact": "9876543210",
     "email": "authenticator@example.com",
     "firstname": "Hritik",
     "lastname": "Khandare",
     "middlename": "Ramesh",
     "location": "Kundevahal"
   }
 }
}
```


Integrity Rules:

- Unique authenticatorId prevents duplication.
- Location is mandatory for tracking item verification.

3. Lost Items Collection

lost_items/{itemId}
Each lost item is stored under a unique itemId (generated via push keys in Firebase).


```
{
 "lost_items": {
   "item001": {
     "adharImageUrl": "https://example.com/image.jpg",
     "adharNumber": "123456789012",
     "dateTime": "2024-01-30T10:30:00Z",
     "description": "Lost wallet near park",
     "finalImageType": "orignal",
     "itemId": "item001",
     "itemImageUrl": "https://example.com/item.jpg"
   }
 }
}
```

Integrity Rules:

- itemId is unique (generated via Firebase push keys).
- Aadhaar number can be verified in the frontend before storing.

4. Accepted Items Collection

accepted_items/{itemId}
Accepted items are stored in a separate collection with an additional location field.


```
{
  "accepted_items": {
    "item001": {
      "adharImageUrl": "https://example.com/image.jpg",
      "adharNumber": "123456789012",
      "dateTime": "2024-01-30T11:00:00Z",
      "description": "Wallet found and verified",
      "finalImageType": "orignal",
      "itemId": "item001",
      "itemImageUrl": "https://example.com/item.jpg",
      "location": "Kundevahal Police Station"
    }
  }
}
```


Integrity Rules:

- itemId should exist in lost_items (not enforced by Firebase but can be validated in the app).
- Location is required for tracking.

5. Handover Records Collection

handover_records/{handoverId}
Once an item is handed over, its details are recorded here.


```
{
  "handover_records": {
```

```
  "handover001": {
    "description": "Black wallet with cash",
    "email": "user@example.com",
    "id": "handover001",
    "itemName": "Wallet",
    "location": "Kundevahal Gram Panchayat",
    "phoneNumber": "9876543210",
    "photoUrl": "https://example.com/photo.jpg",
    "username": "Jatin"
  }
 }
}
```


Integrity Rules:

- handoverId is unique (generated using Firebase push keys).
- email should match a user's email in Users, ensuring only registered users can collect items.

How Firebase Ensures Data Integrity

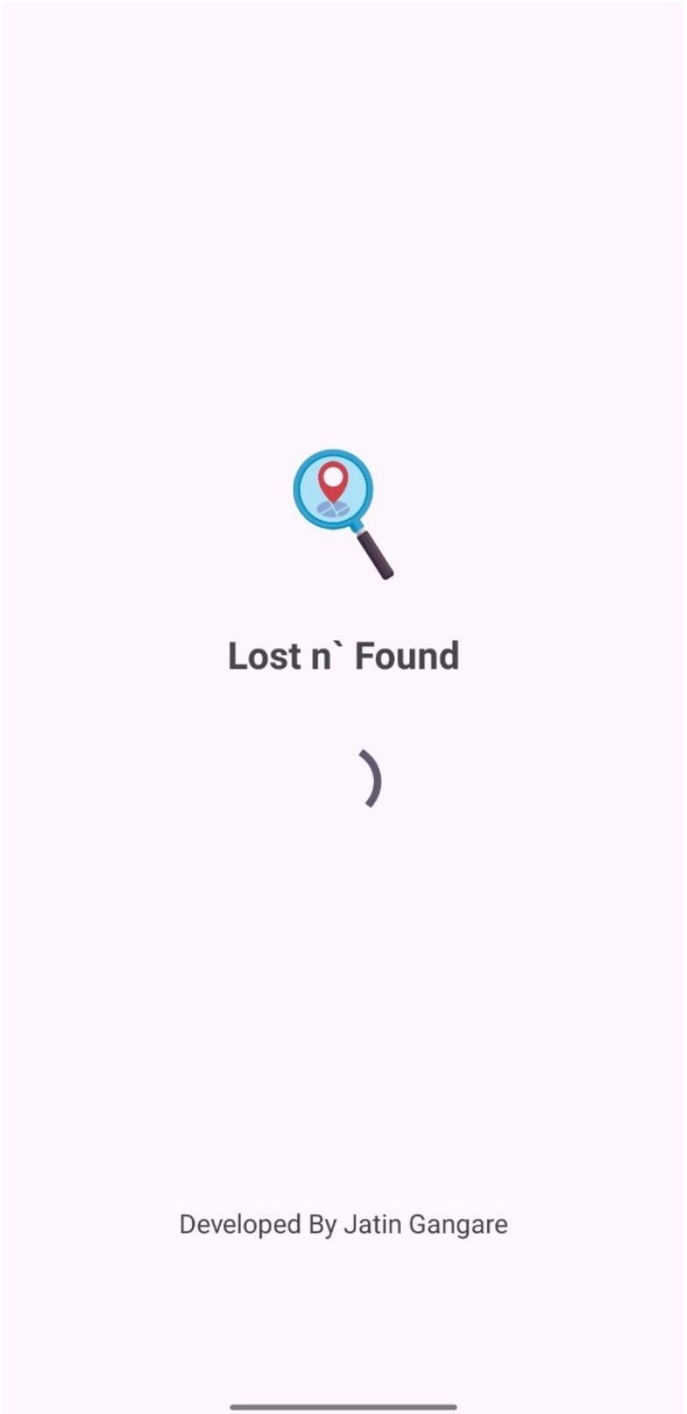
Since **NoSQL** databases like Firebase don't enforce constraints like primary keys or foreign keys, we maintain integrity through:

1. **Unique IDs**: Firebase auto-generates unique keys for each record.
2. **Firebase Authentication**: Ensures only registered users can log in.
    a. **Validation Rules** (Firebase Database Rules):
    b. **Ensure unique emails** using Firebase Authentication.
    c. **Restrict data modification** (e.g., only an Authenticator can accept lost items).
3. **App-Level Validations**:
    a. Check if an itemId exists before adding it to accepted_items.
    b. Verify email matches user record before allowing handover.

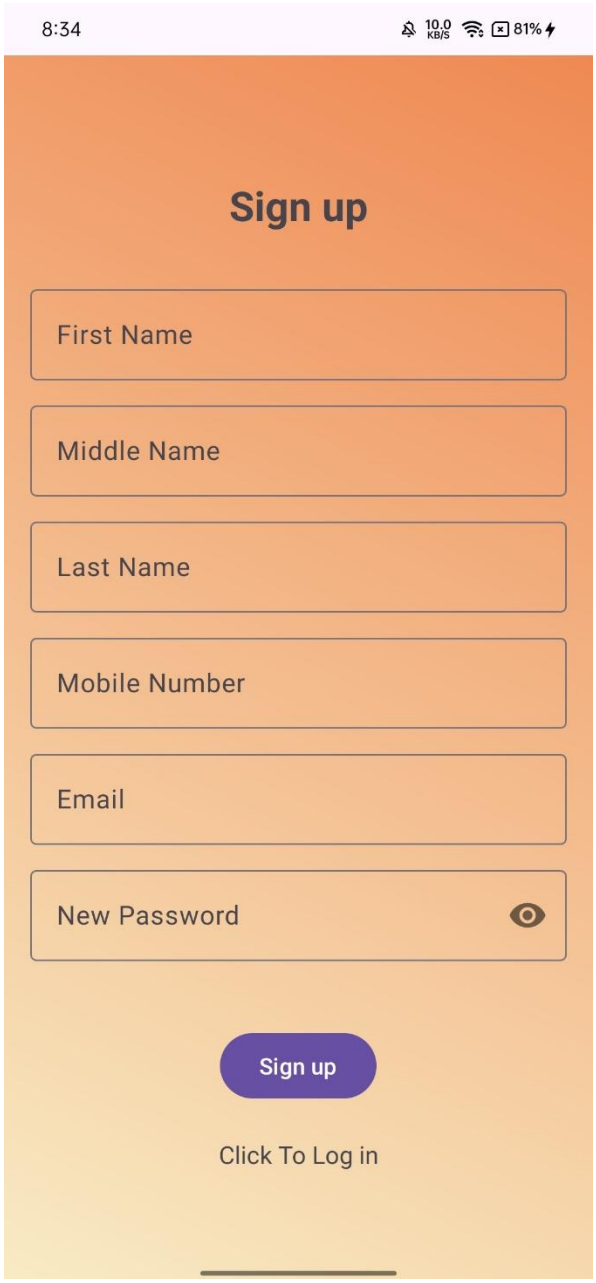
4.3 User Interface and Design

The User Interface (UI) Design of the *Lost n' Found* system focuses on creating a user-friendly and visually appealing experience for both users and authenticators. The design follows a simple and intuitive layout, ensuring easy navigation and accessibility.

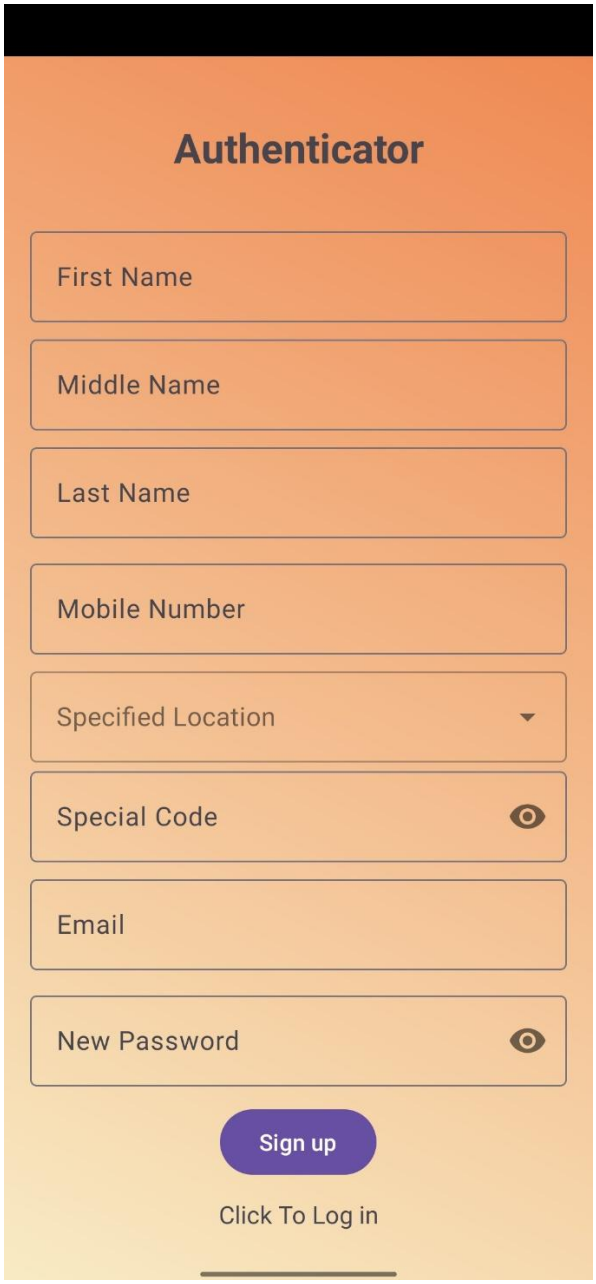
Splash Screen :



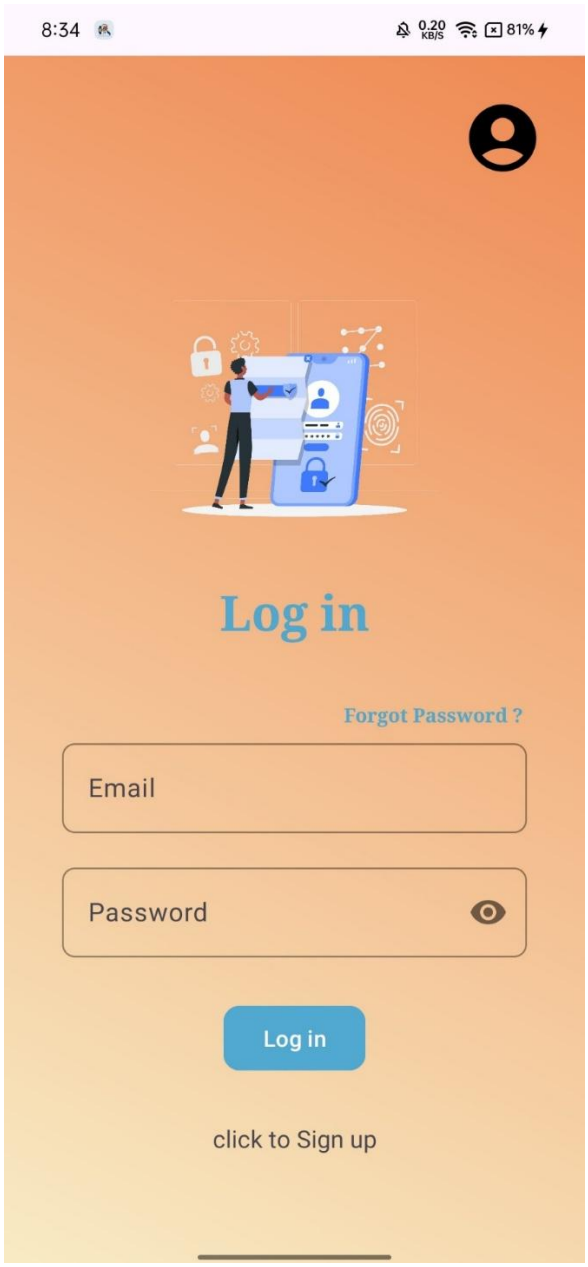
User SignUp :



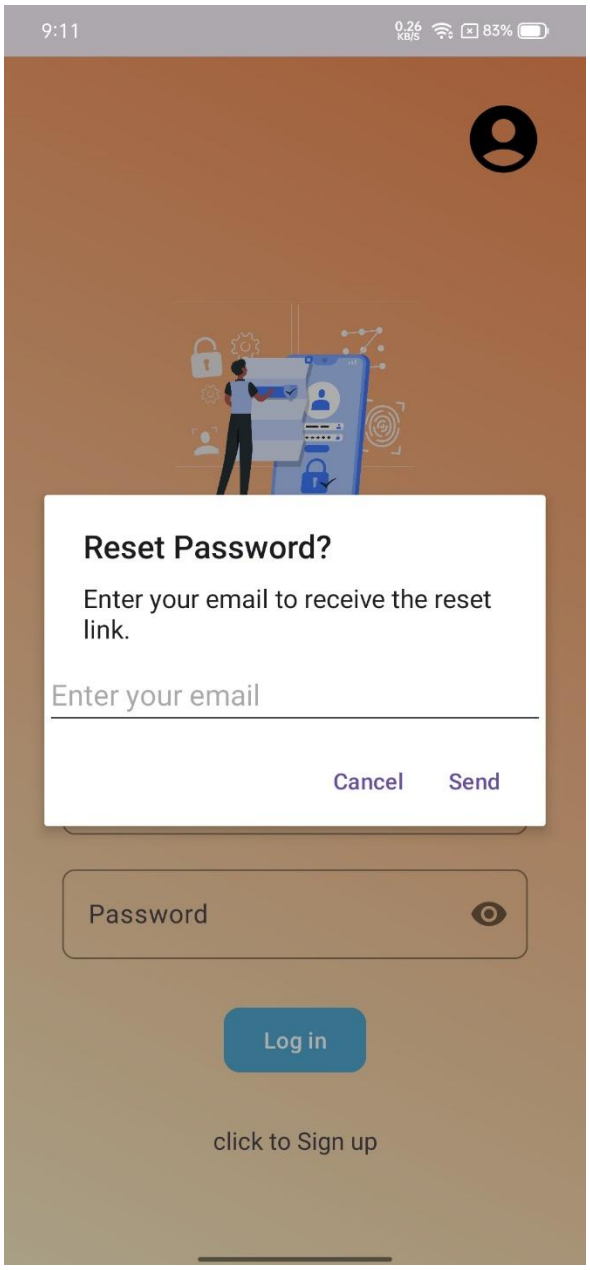
Authenticator Signup :



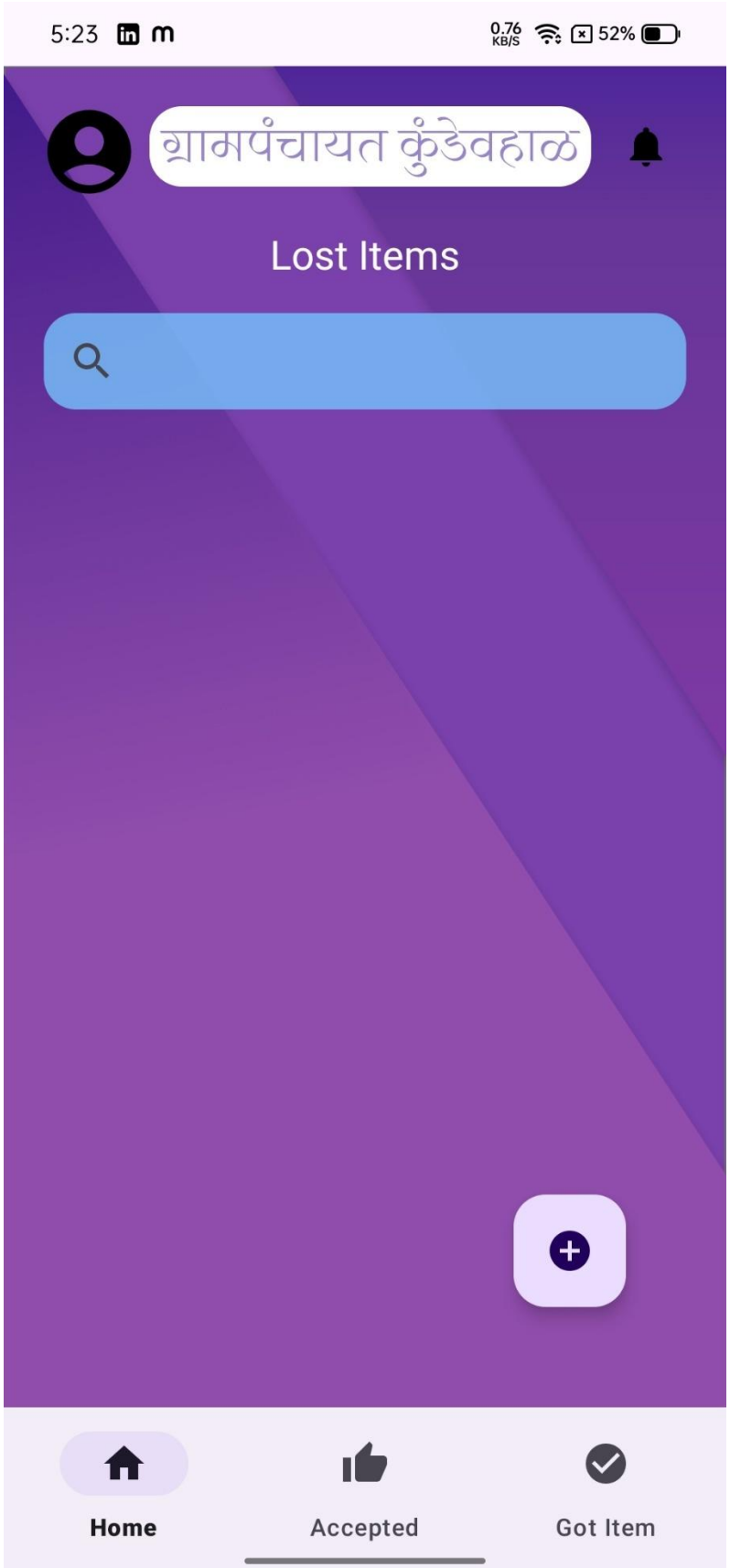
Login :



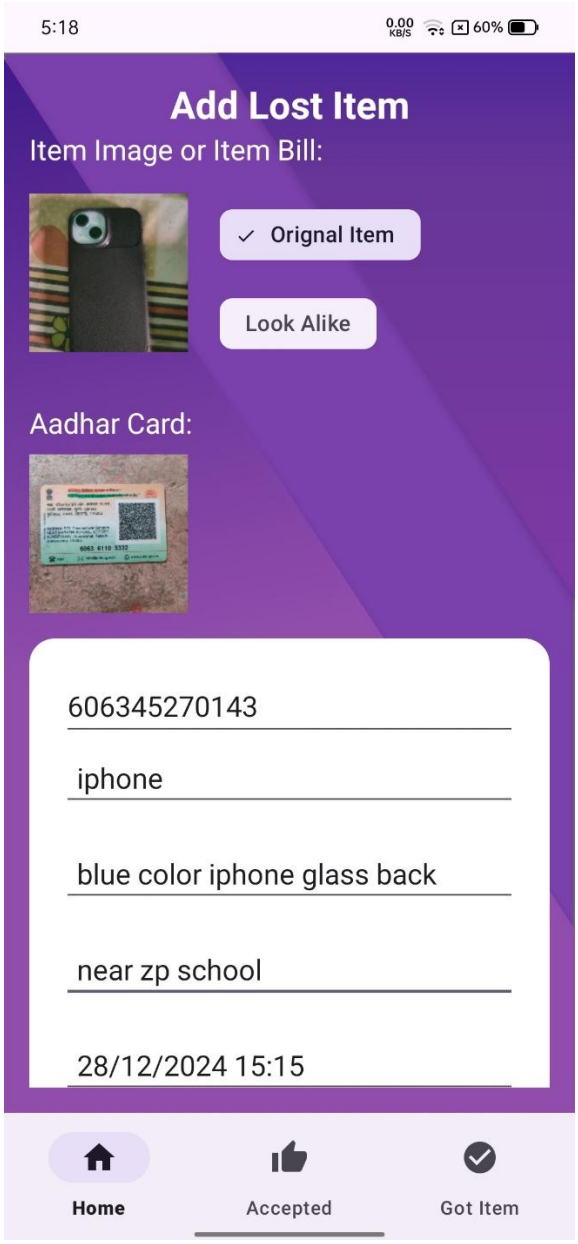
Forgot Password :



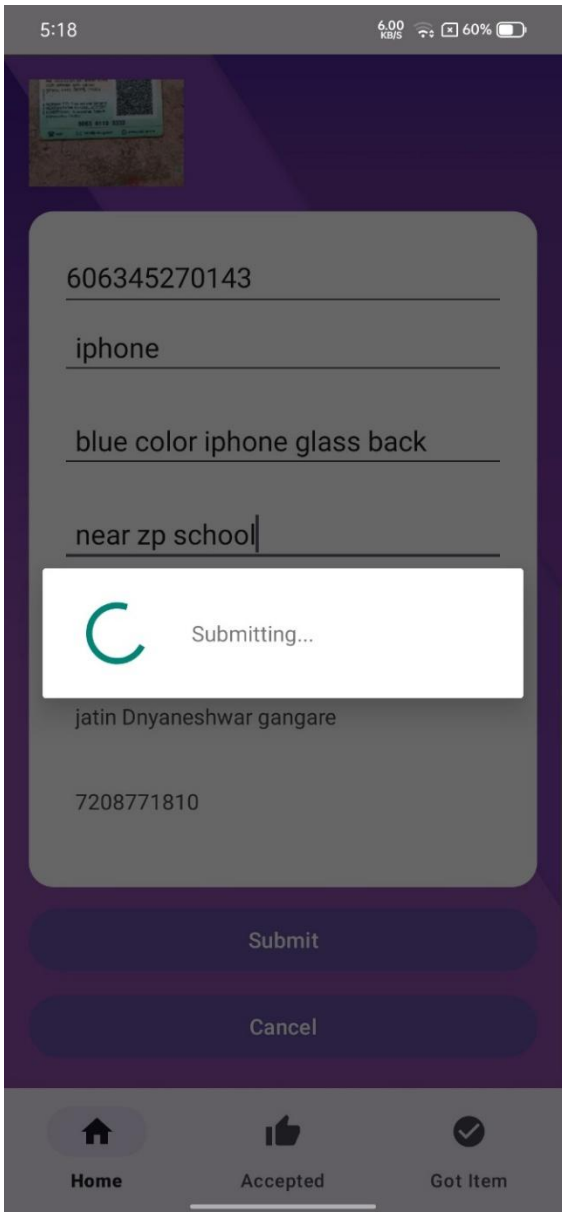
User Dashboard :



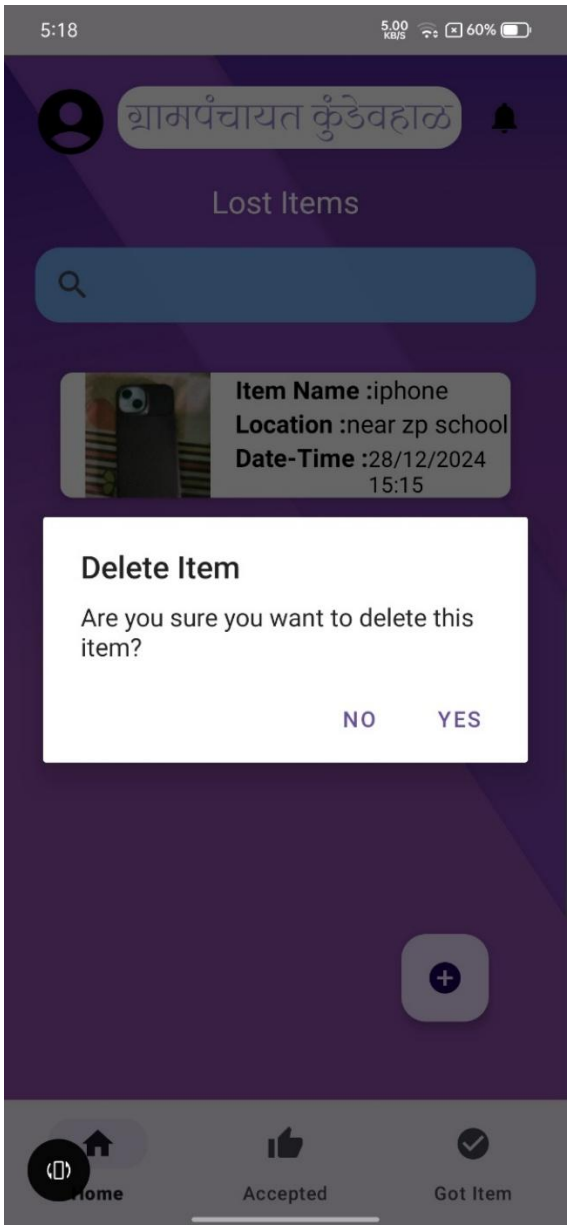
Lost Item Submission :



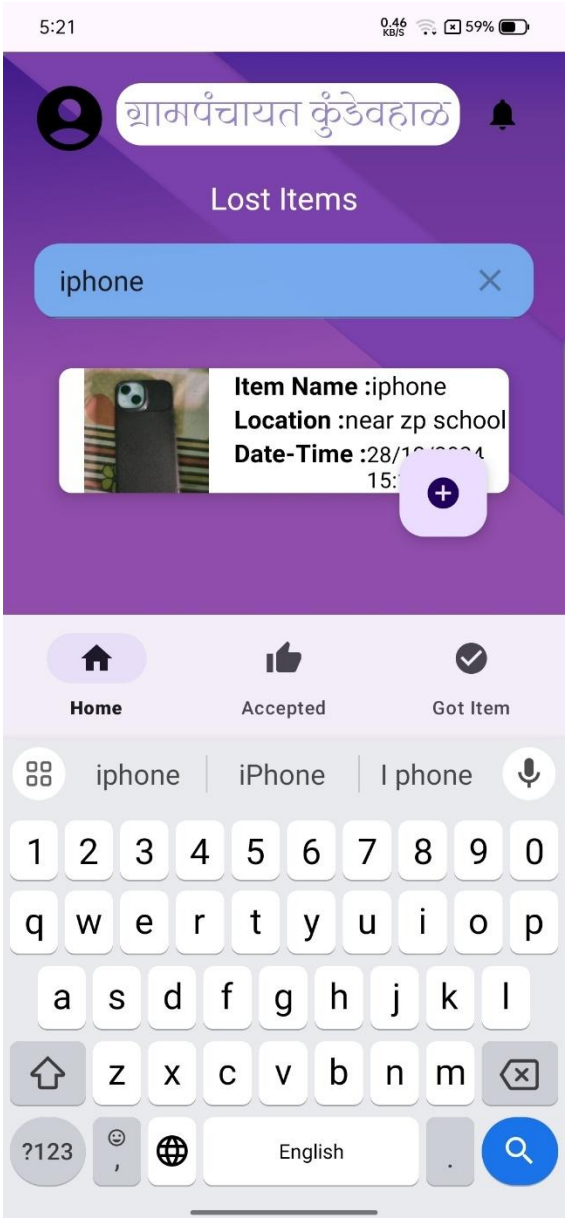
After Lost Item Submission :



Item Deletion Operation :

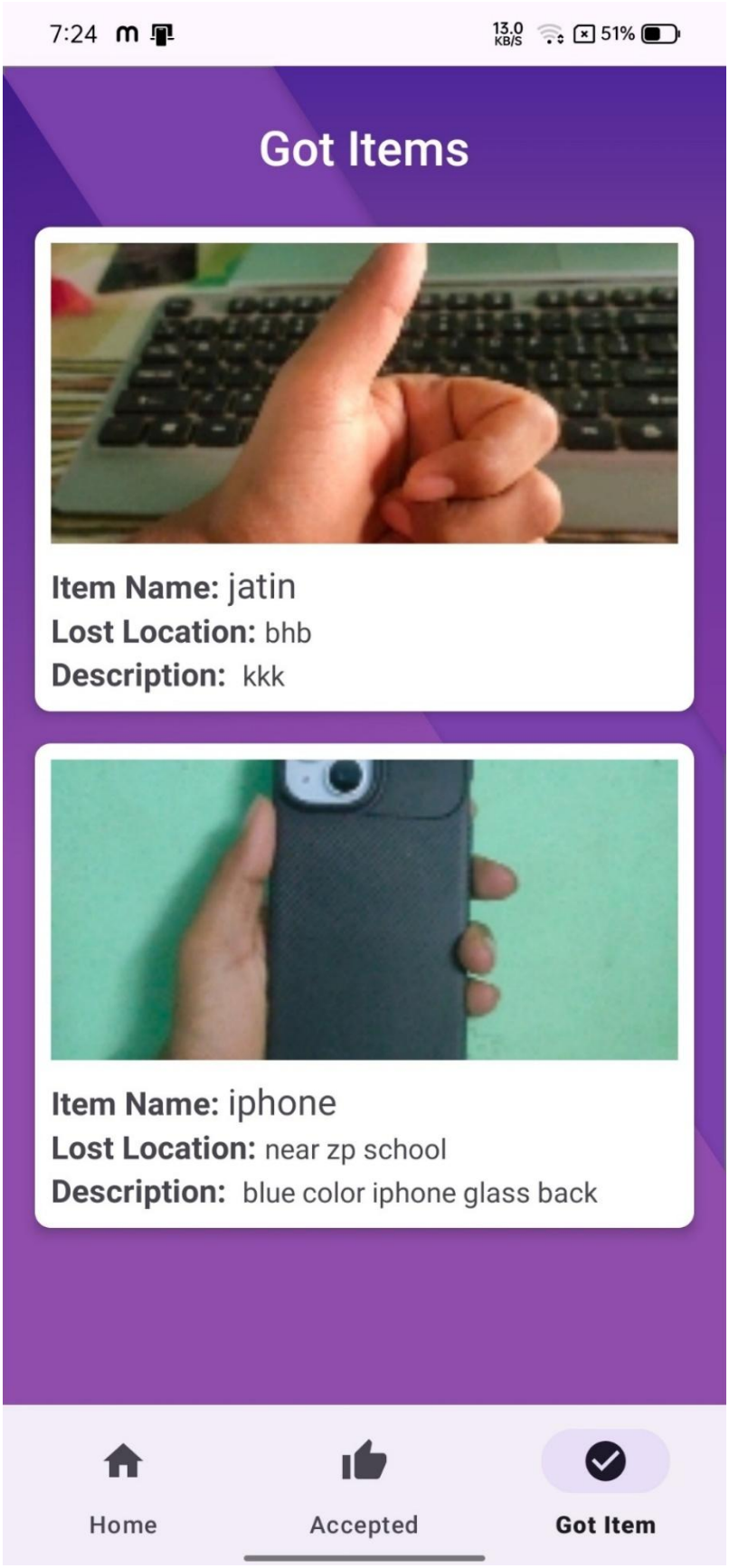


Search Operation :

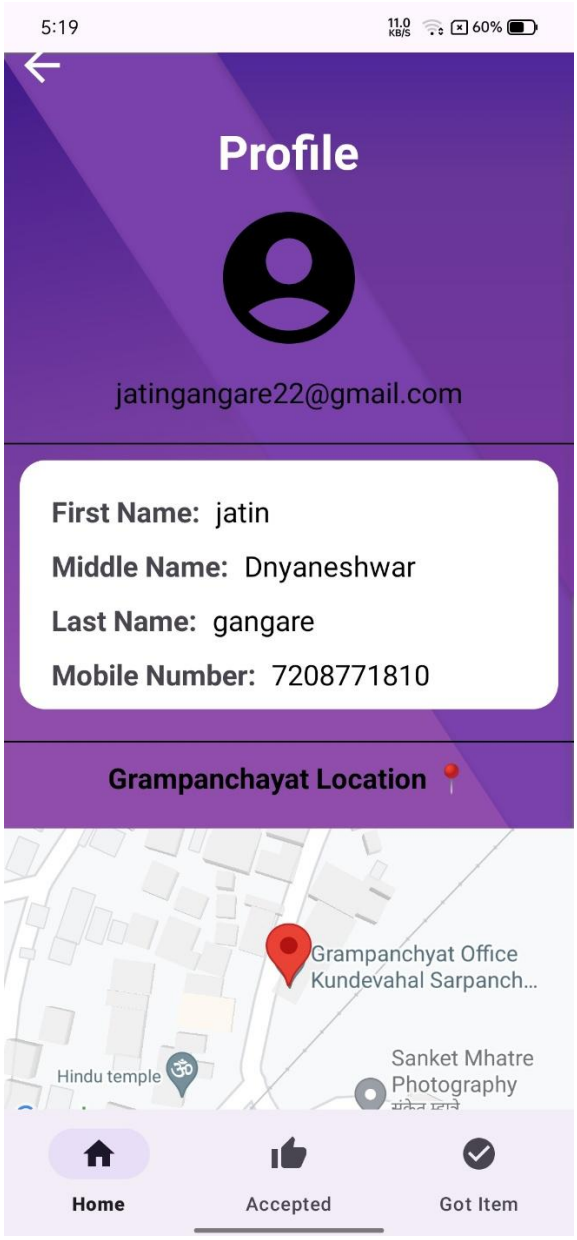


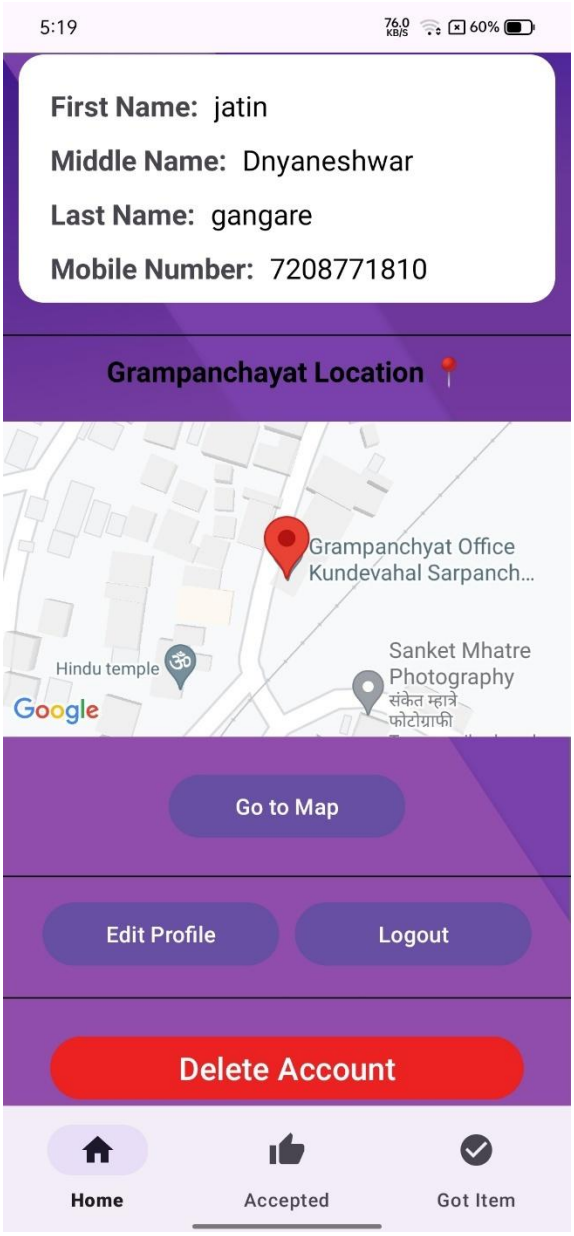
Accepted Item (User) :

Got Item (User) :

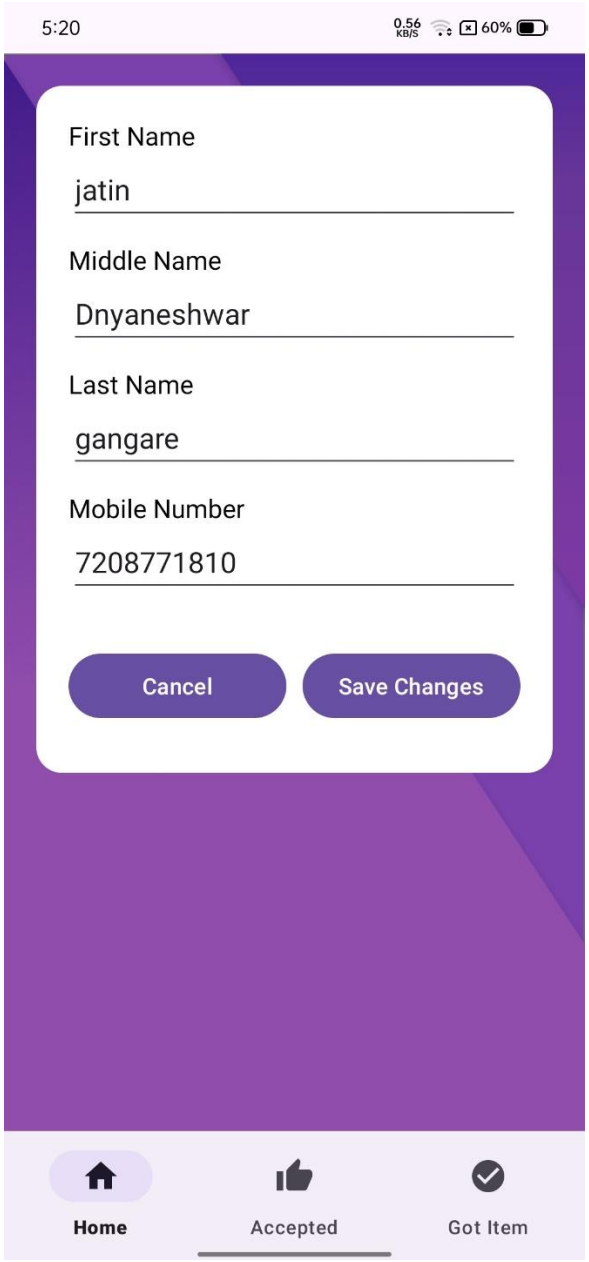


User Profile :

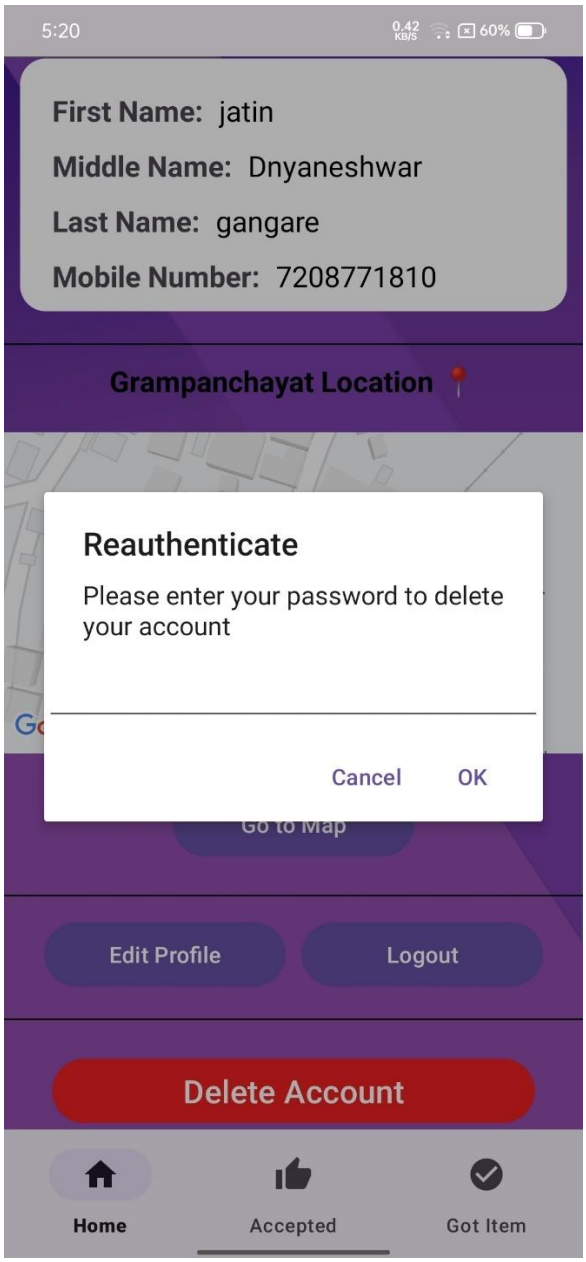




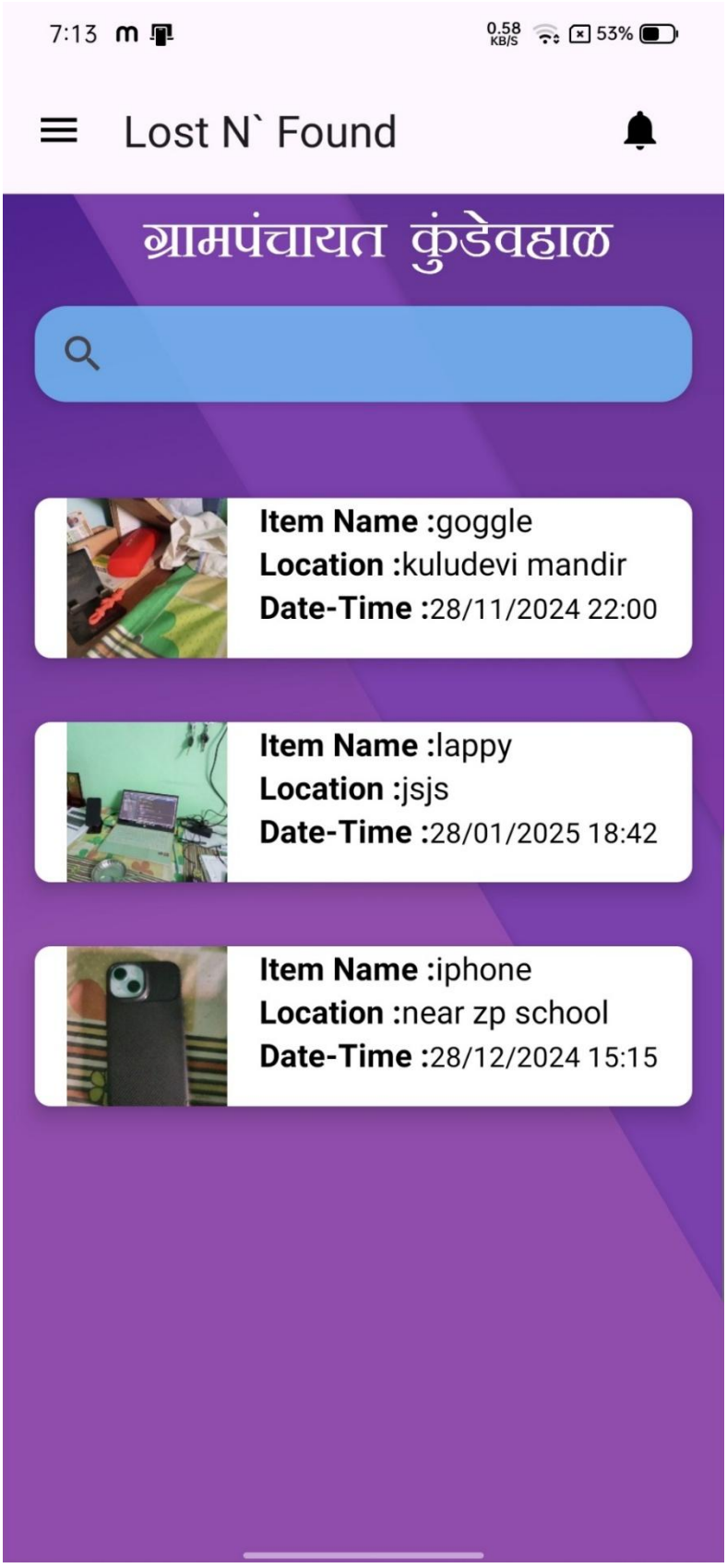
Profile Updation (User) :



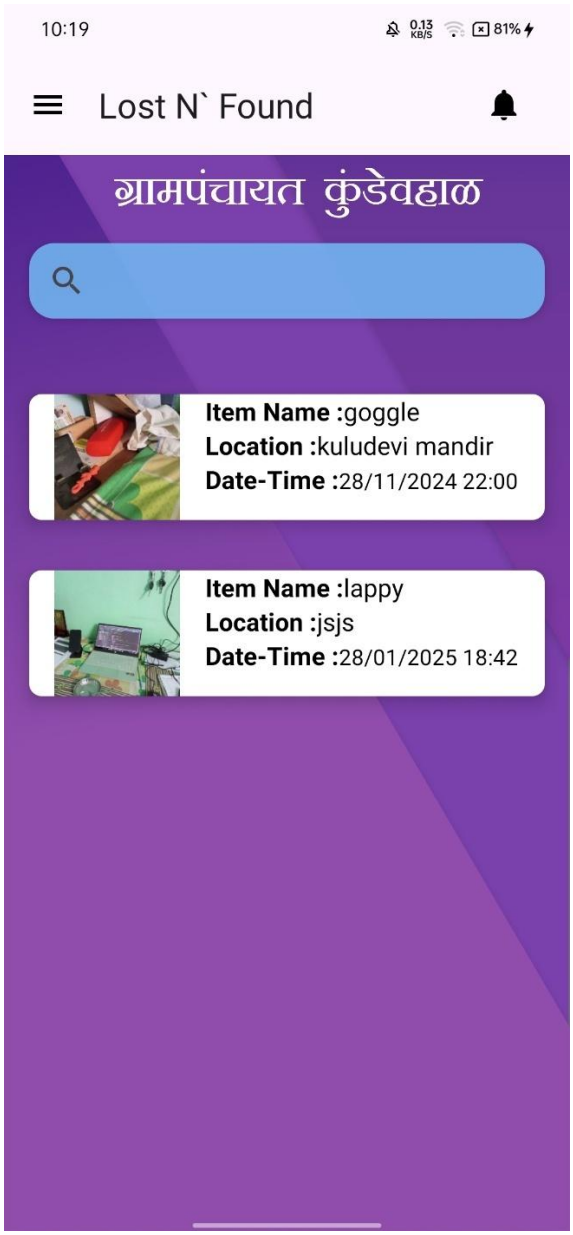
Account Deletion (User) :

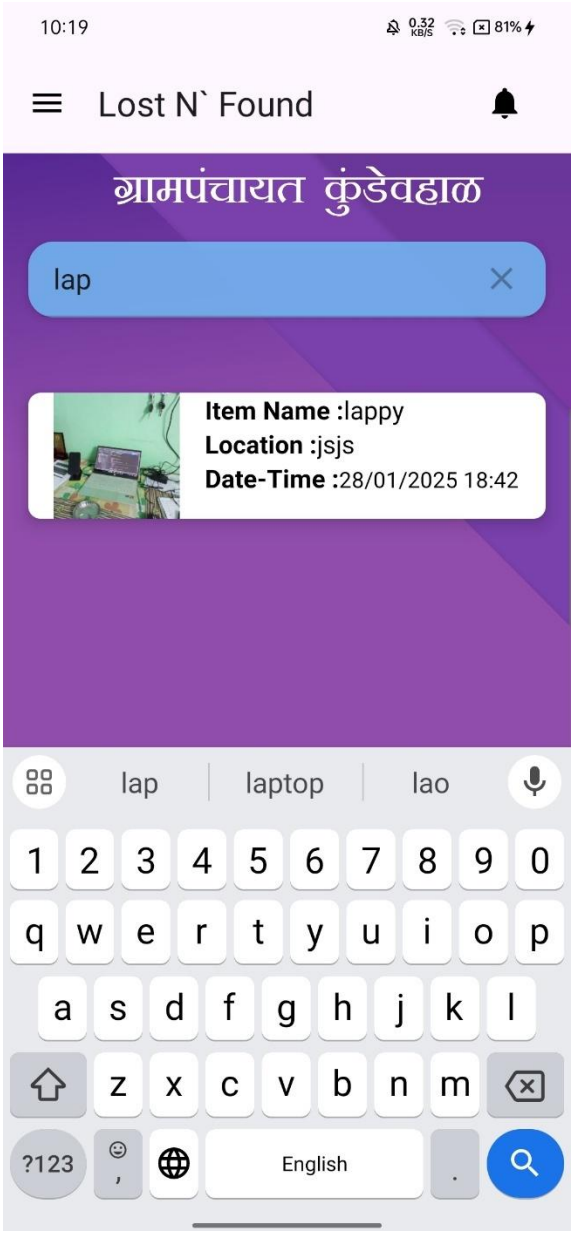


Authenticator Dashboard :

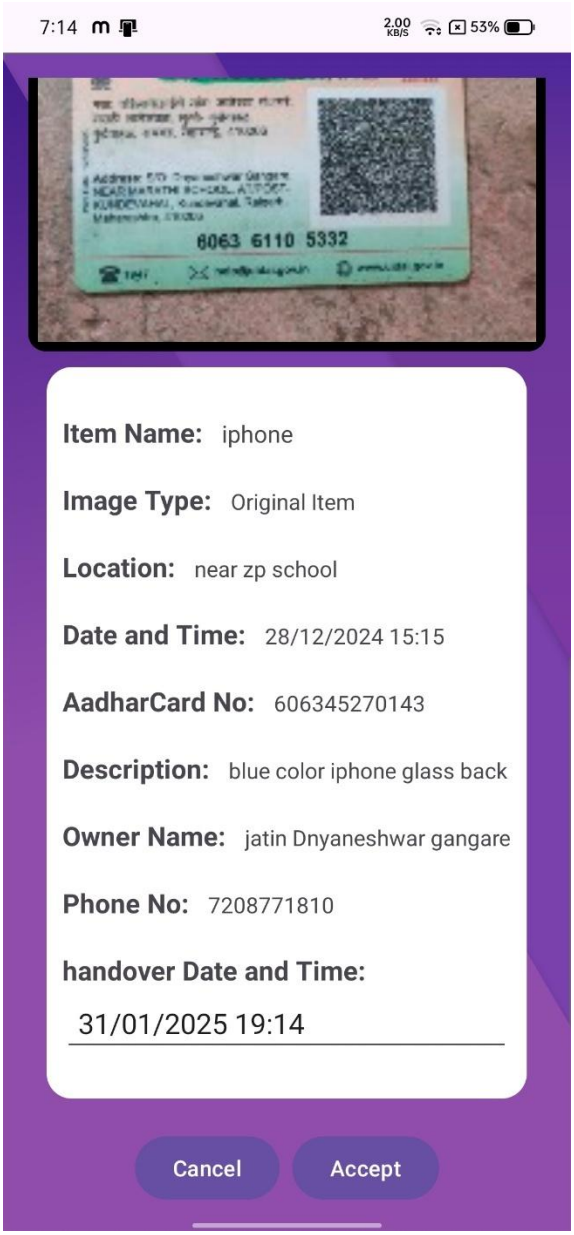


Search Operation :

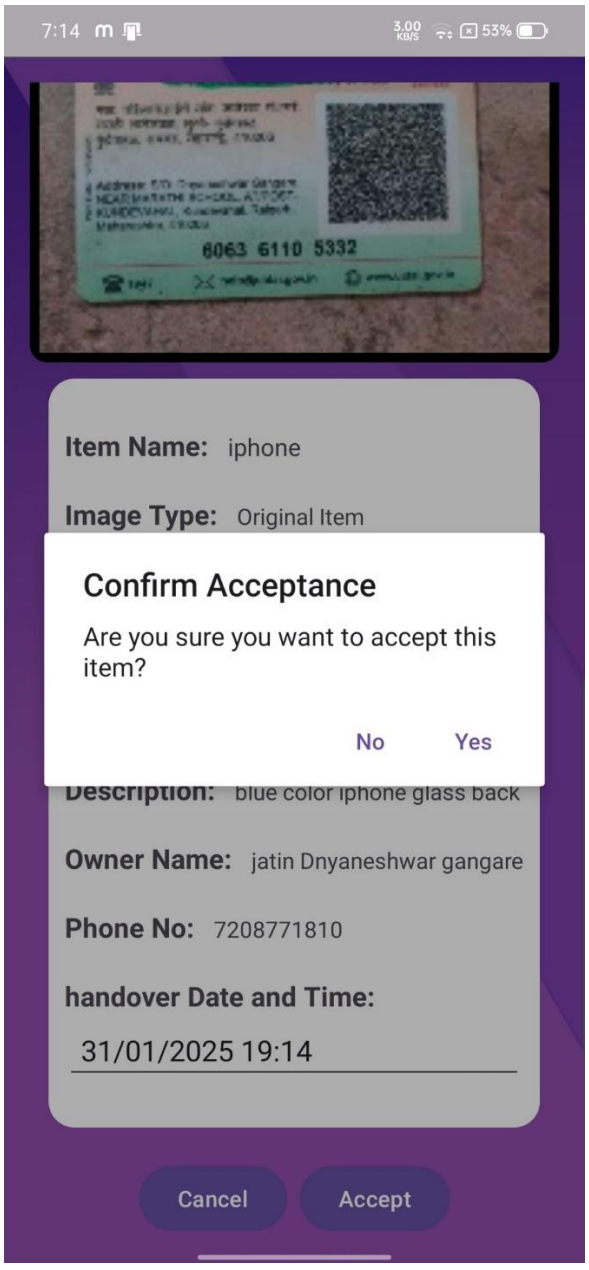


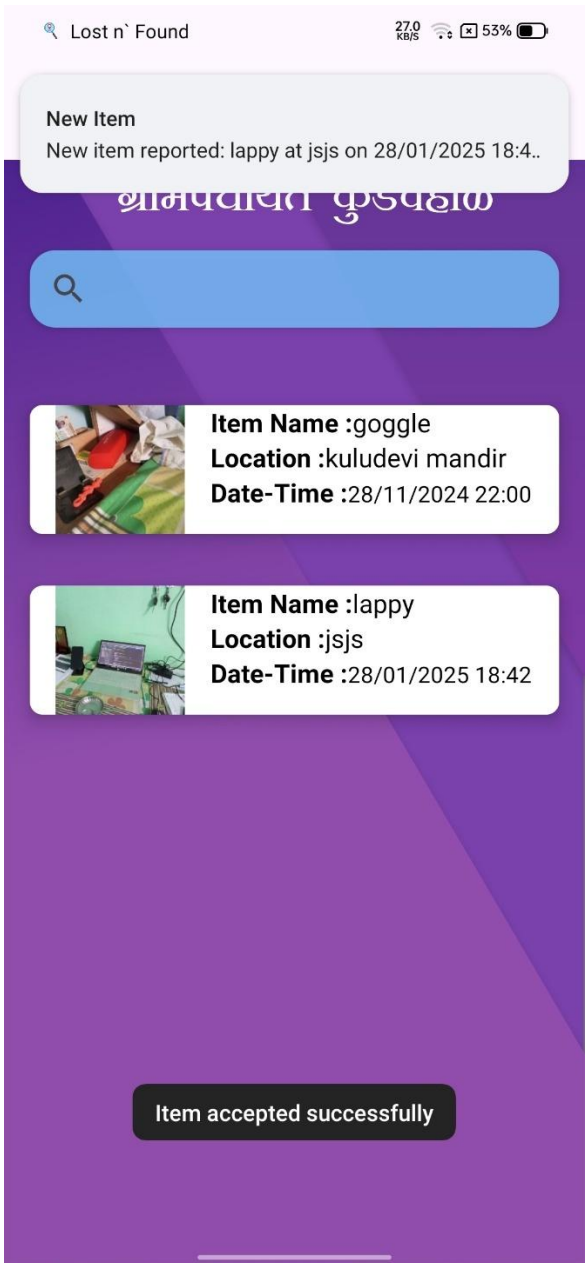


Item Verification :

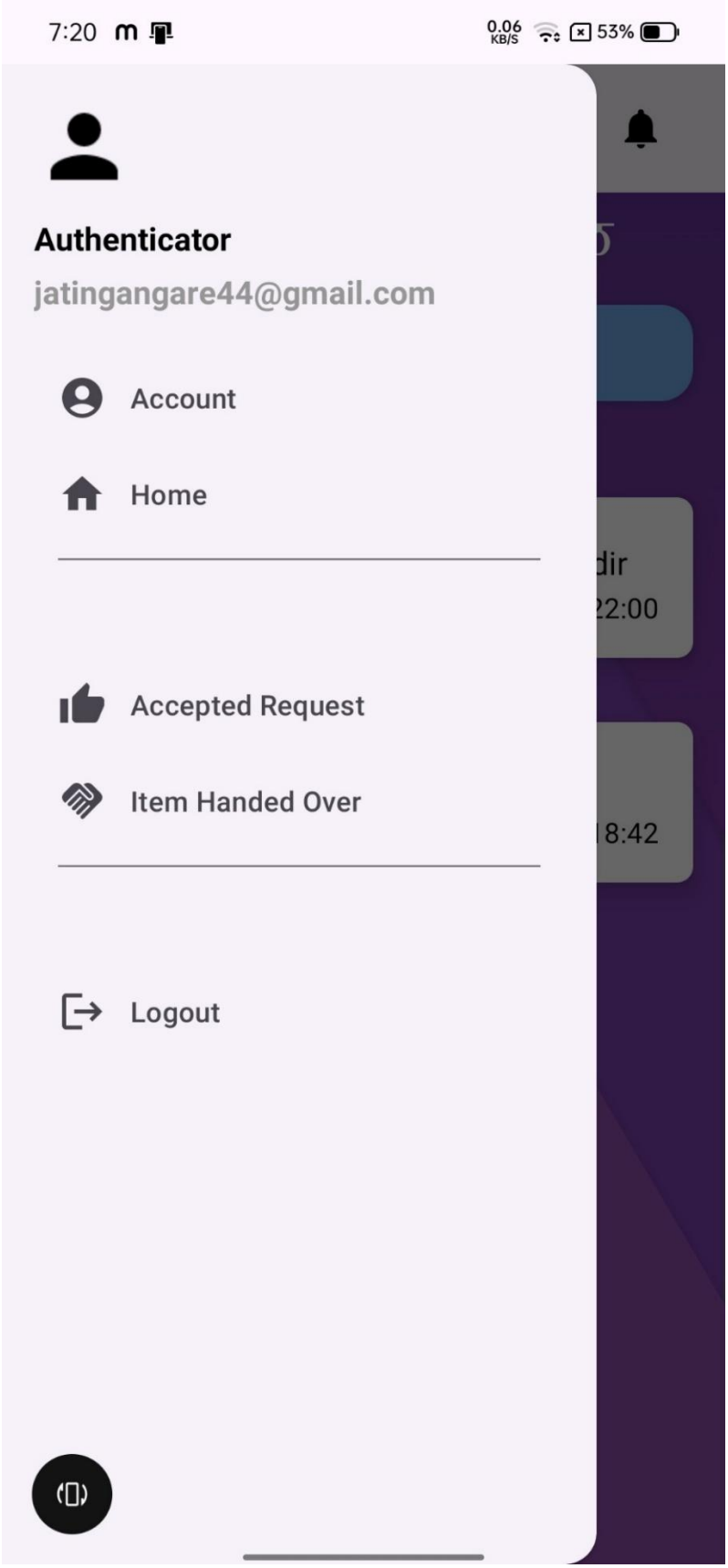


Confirm Acceptance :

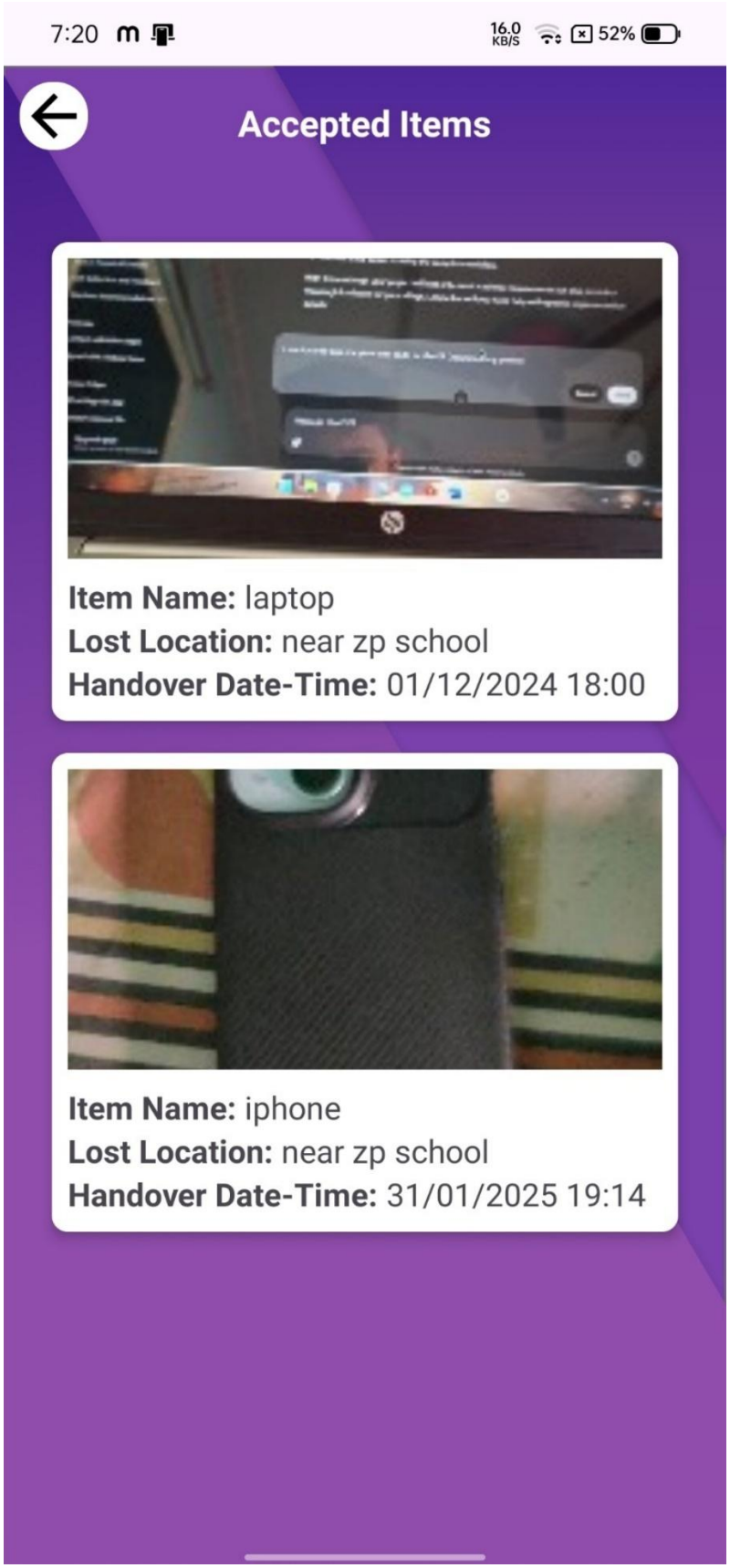




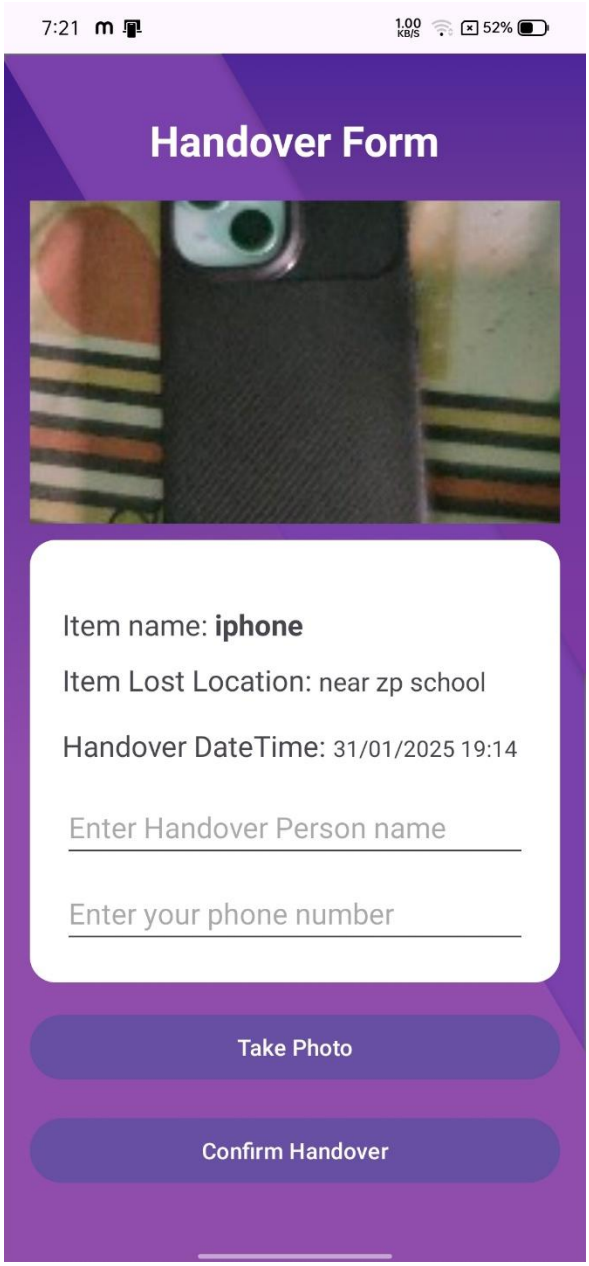
Navigation Panel (Authenticator) :

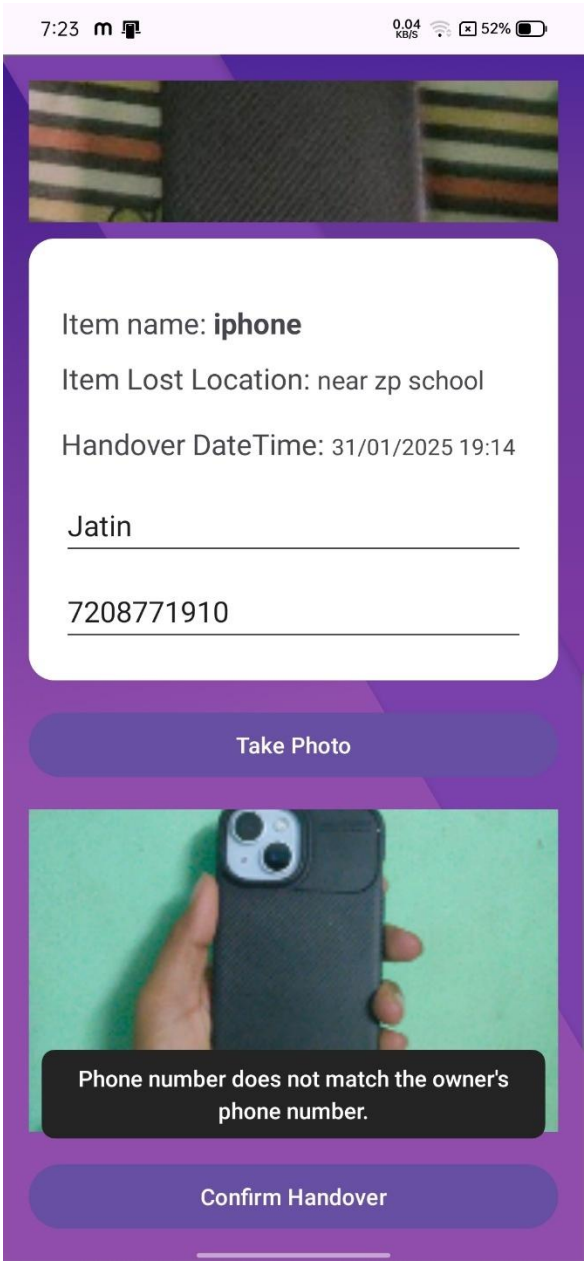


Accepted Item List (accepted request from Navigation Panel) :

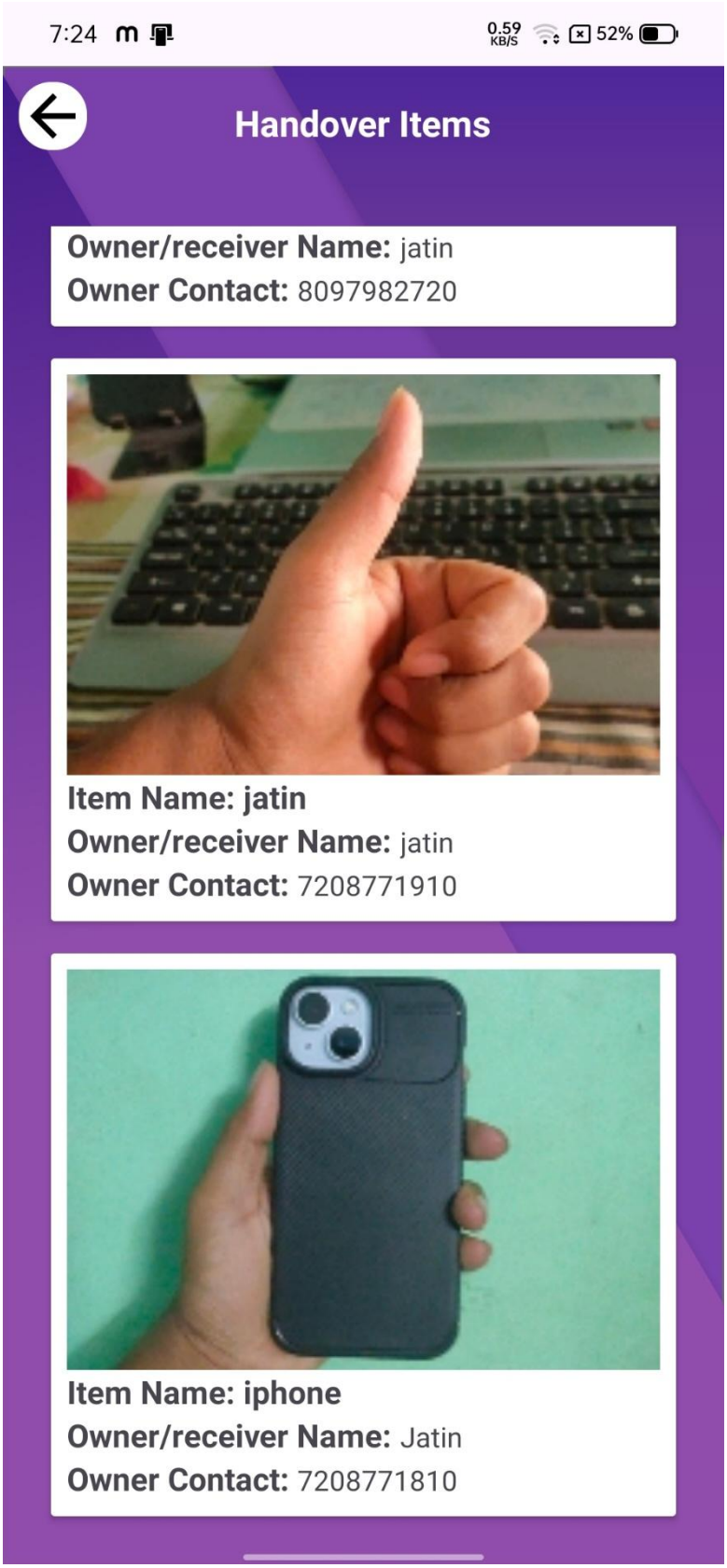


Handing Over Verification (Handover Form):

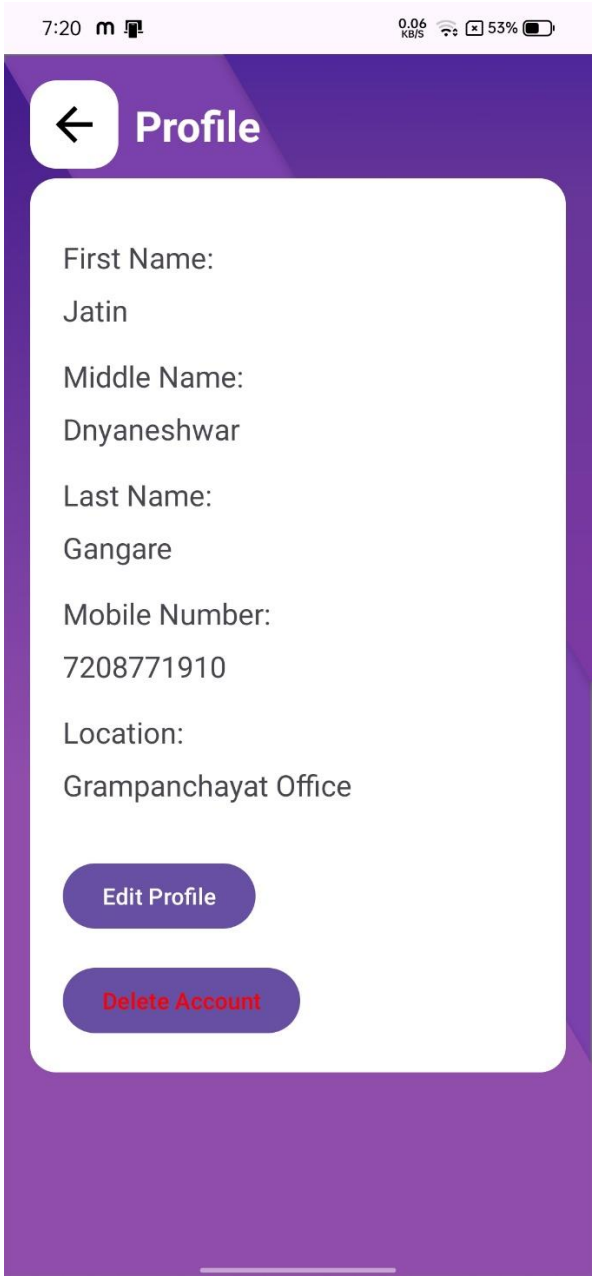




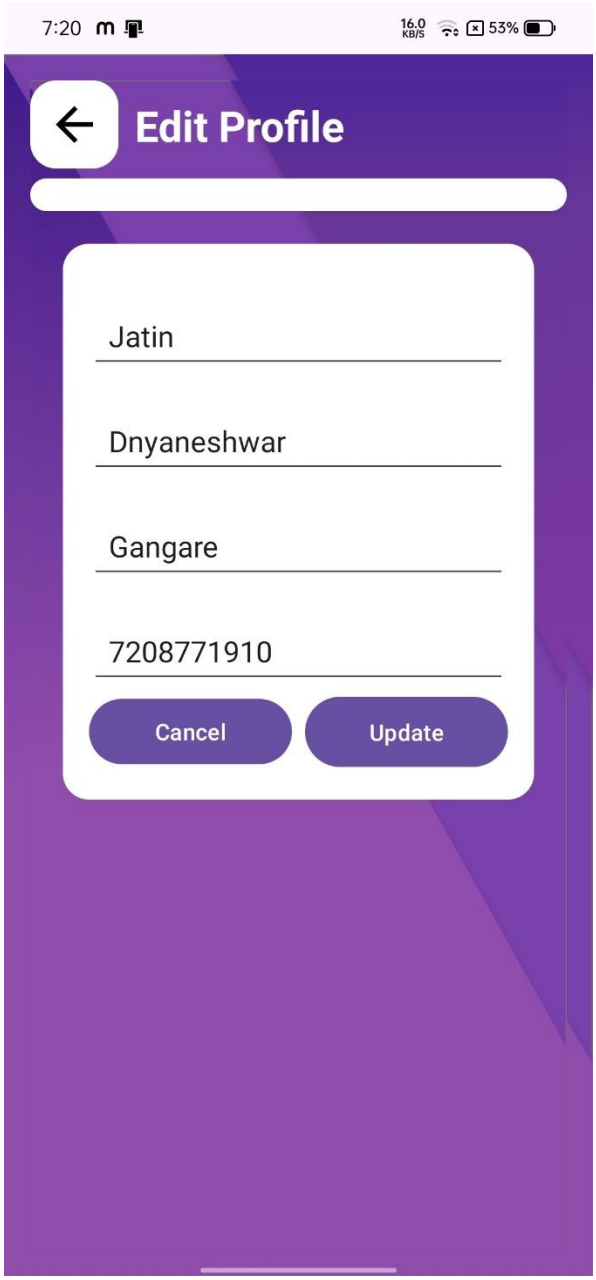
Handed Over List :



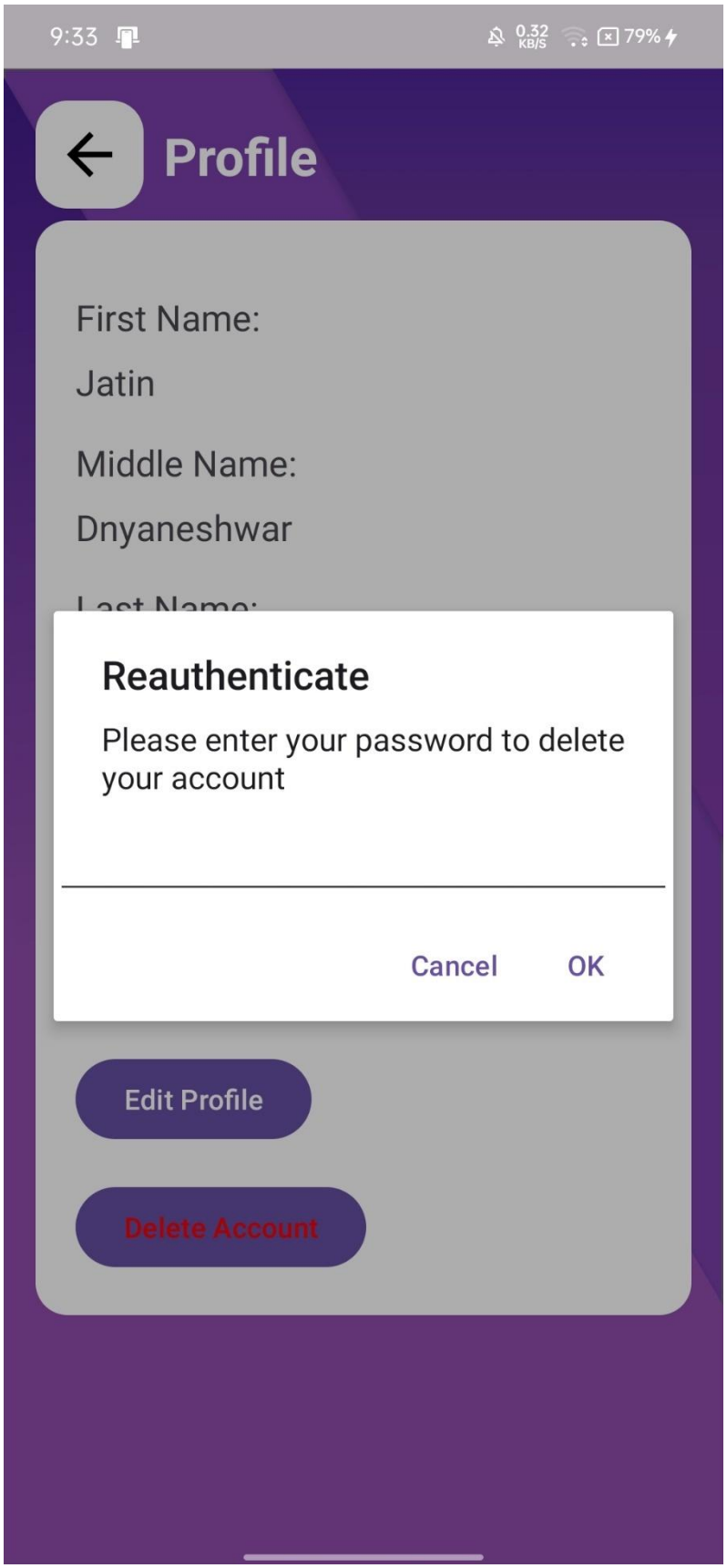
Authenticator Profile :



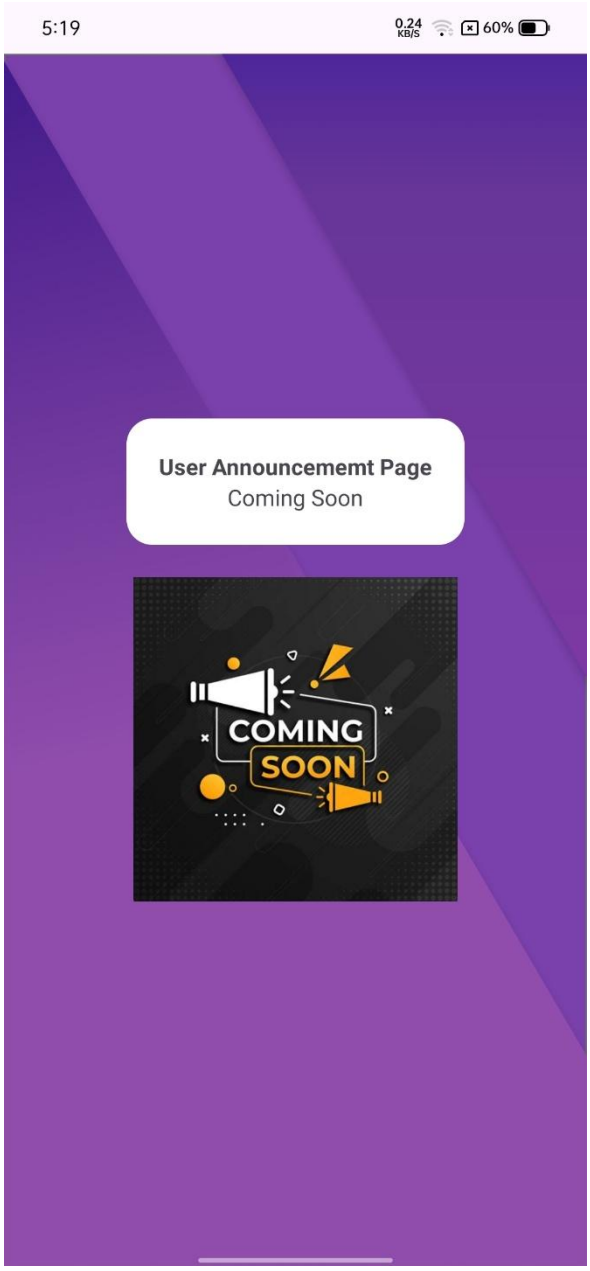
Updating Profile :

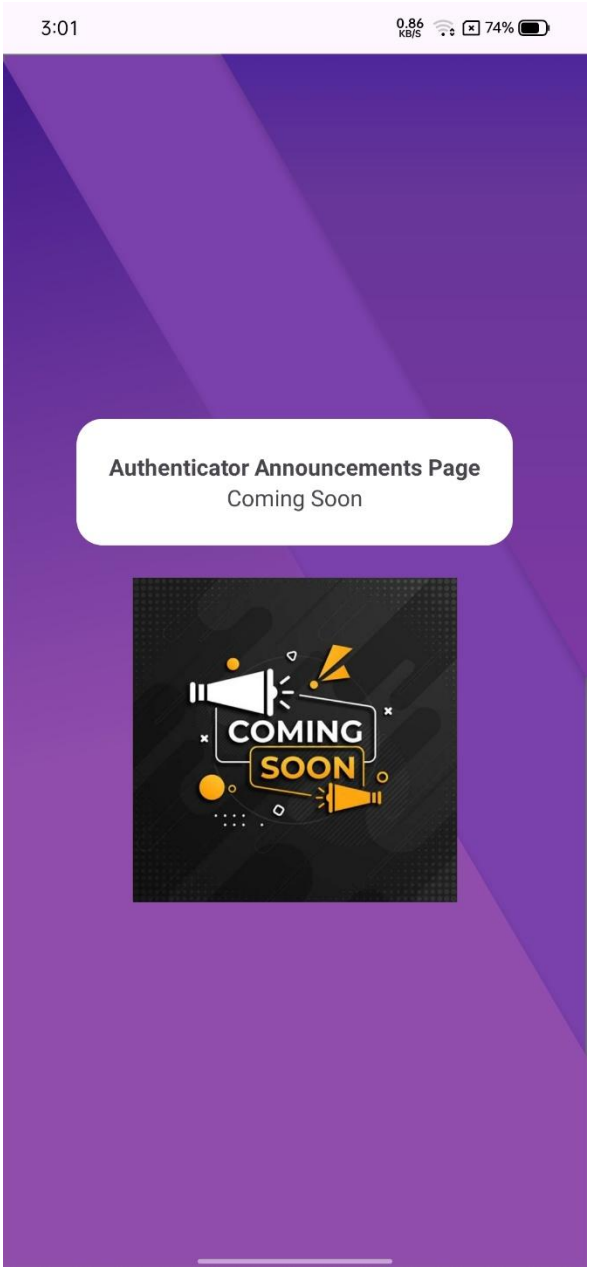


Authenticator Account Deletion :



User and Authenticator Announcement Screen (Future Update)





4.4 Security Issues

Security is a crucial aspect of the *Lost n' Found* system, as it handles sensitive user data, including personal information, lost item details, and authentication credentials. The application ensures security through multiple layers of protection, such as authentication, database security, and encrypted file storage. Below are the major security concerns and the measures implemented to mitigate them.

1. Authentication & Authorization

Authentication and authorization ensure that only legitimate users can access the system. The application uses Firebase Authentication, which supports email/password login. User credentials are securely stored and managed by Firebase, reducing the risk of password leaks. Additionally, Firebase Authentication enables multi-factor authentication (MFA), adding an extra layer of security by requiring an additional verification step, such as OTP (One-Time Password). Role-based access control (RBAC) ensures that authenticators have special privileges, such as approving lost item claims, while regular users can only submit and search for lost items. This prevents unauthorized users from manipulating the system.

2. Data Privacy & Protection

Since the system stores personally identifiable information (PII) such as Aadhaar details, email addresses, and contact numbers, it is essential to maintain privacy. Firebase Realtime Database encrypts data in transit using SSL/TLS (Secure Socket Layer/Transport Layer Security), ensuring that sensitive information is not exposed during data transmission. Additionally, Firebase security rules restrict access to user data, allowing only authorized users to view or modify their records. User data is not publicly accessible, and strict permission controls are applied to prevent data breaches.

3. Database Security

Firebase Realtime Database is structured as a NoSQL database, which requires special security configurations to prevent unauthorized access. Database security rules are implemented to define which users or roles can read or write specific data. For example, only the user who submitted a lost item can view or edit its details, and only an authenticator can approve item claims. These rules prevent direct database manipulation and unauthorized modifications.

Additionally, access tokens are used to verify user identity before allowing database interactions, reducing the risk of unauthorized access.

4. Secure File Storage

The *Lost n' Found* system allows users to upload images, such as Aadhaar images for verification and item photos. Firebase Storage is used to manage these files securely. Each uploaded file is assigned a unique identifier, and download URLs are restricted to prevent public access. Users must be authenticated to retrieve their uploaded images, ensuring that private data remains confidential. Furthermore, Firebase Security Rules enforce controlled access to storage, ensuring that only the file owner or relevant authenticator can access specific files.

5. Protection Against NoSQL Injection

Unlike traditional SQL databases, Firebase is a NoSQL database, making it susceptible to NoSQL injection attacks, where malicious users attempt to manipulate data queries. To prevent this, the system validates and sanitizes user input before storing it in the database. Additionally, Firebase security rules enforce strict access policies, ensuring that unauthorized data modifications are blocked. The application follows the principle of least privilege, meaning users can only access the minimum amount of data necessary for their role.

6. Preventing Data Tampering

One of the security risks in the *Lost n' Found* system is the possibility of users tampering with data to falsely claim lost items. To prevent this, a multi-step verification process is implemented. Users must submit Aadhaar verification along with item details when claiming an item. Authenticators are the only entities authorized to verify and approve claims. Once a handover is completed, the data is logged in a secure handover_records section of the database, preventing further modifications. Any suspicious activity is logged, ensuring system administrators can review and take necessary actions.

4.5 Test Cases

Test Case ID	Test Scenario	Test Steps	Expected Output	Actual Output	Status (Pass/Fail)
TC_01	User Registration	1. Open the app. 2. Navigate to the SignUp page. 3. Enter valid details (name, email, password). 4. Click "SignUp".	Account successfully created, user redirected to login page.	Account successfully created, user redirected to login page.	Pass
TC_02	User Login	1. Open the app. 2. Enter valid email and password. 3. Click "Login".	User successfully logs in and navigates to the dashboard.	User successfully logs in and navigates to the dashboard.	Pass
TC_03	Invalid Login	1. Open the app. 2. Enter incorrect credentials. 3. Click "Login".	"Invalid email or password" message displayed.	"Invalid email or password" message displayed.	Pass
TC_04	Lost Item Submission	1. Log in as a user. 2. Click on “+” icon to list Lost Item. 3. Fill in item details and upload Aadhaar image. 4. Click "Submit".	Lost item successfully submitted, notification sent to authenticator.	Lost item successfully submitted, notification sent to authenticator.	Pass
TC_05	Search Lost Items	1. Log in as a user. 2. Navigate to the search bar. 3. Enter item name or description. 4. Click "Search icon".	Matching lost items are displayed.	Matching lost items are displayed.	Pass
TC_06	Authenticator Approval	1. Log in as an authenticator. 2. Open lost item requests. 3. Verify details and approve request or ignore.	Item moved to "Accepted Items", notification sent to user.	Item moved to "Accepted Items", notification sent to user.	Pass

TC_07	Handover Process	1. User meets authenticator. 2. Authenticator verifies user identity. 3.Completes handover form. 4.Clicks "Confirm Handover".	Database updated, notifications sent.	Database updated, notifications sent.	Pass
TC_08	Unauthorized Access	1. Click on Authenticator SignUp 2. Try Signing up without Knowning /invalid Special Key.	“Invalid Special Code” message displayed.	“Invalid Special Code” message displayed.	Pass
TC_09	Password Reset	1. Open the app. 2. Navigate to the "Forgot Password" option on the login screen. 3. Enter a registered email address. 4. Click on "Reset Password." 5. Check the email for a password reset link and attempt to reset the password.	A password reset link should be sent to the entered email, allowing the user to create a new password.	A password reset link was successfully sent, and the user was able to reset their password.	Pass
TC_10	File Upload Security	1. Upload a non-image file as an item image.	System rejects invalid file format.	System rejects invalid file format.	Pass

CHAPTER 5

SYSTEM CODING, IMPLEMENTATION AND TESTING

5.1 Coding Details

5.2 Coding Efficiency

Splash Screen :

1. SplashActivity.java

```
package com.kundevahal.lostnfound;

public class SplashActivity extends AppCompatActivity {
    private static final String TAG = "SplashActivity";
    Handler handler;
    FirebaseAuth mAuth;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        this.getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);
        getSupportActionBar().hide();
        setContentView(R.layout.activity_splash);
        mAuth = FirebaseAuth.getInstance();

        handler = new Handler();
        handler.postDelayed(new Runnable() {
            @Override
            public void run() {
                FirebaseUser currentUser = mAuth.getCurrentUser();
                if (currentUser != null && currentUser.isEmailVerified()) {
                    checkUserTypeAndRedirect(currentUser.getEmail());
                } else {
                    Intent intent = new Intent(SplashActivity.this, LoginActivity.class);
                    startActivity(intent);
                    finish();
                }
            }
        }, 3000); // 3 seconds delay
    }

    private void checkUserTypeAndRedirect(String email) {
        if (!isNetworkAvailable()) {
            Toast.makeText(this, "No internet connection.", Toast.LENGTH_SHORT).show();
```

```
            return;
        }

        DatabaseReference usersRef = FirebaseDatabase.getInstance().getReference("Users");
        DatabaseReference authsRef =
FirebaseDatabase.getInstance().getReference("Authenticator");

        // Check in Users node
        usersRef.orderByChild("email").equalTo(email).addListenerForSingleValueEvent(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                if (snapshot.exists()) {
                    // If email exists in Users node, redirect to UserMainActivity
                    Intent intent = new Intent(getApplicationContext(), UserMainActivity.class);
                    startActivity(intent);
                    finish();
                } else {
                    // If email doesn't exist in Users node, check in Authenticator node

authsRef.orderByChild("email").equalTo(email).addListenerForSingleValueEvent(new
ValueEventListener() {
                        @Override
                        public void onDataChange(@NonNull DataSnapshot snapshot) {
                            if (snapshot.exists()) {
                                // If email exists in Authenticator node, redirect to
AuthenticatorMainActivity
                                Intent intent = new Intent(getApplicationContext(),
AuthenticatorMainActivity.class);
                                startActivity(intent);
                                finish();
                            } else {
                                // If email doesn't exist in either node
                                Toast.makeText(SplashActivity.this, "User type not recognized.",
Toast.LENGTH_SHORT).show();
                            }
                        }}
            }
    }

    private boolean isNetworkAvailable() {
        ConnectivityManager connectivityManager = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo activeNetworkInfo = connectivityManager.getActiveNetworkInfo();
        return activeNetworkInfo != null && activeNetworkInfo.isConnected();
    }
}
```

2. activity_splash.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SplashActivity"
    android:orientation="vertical"
    android:gravity="center">


    <ImageView
        android:id="@+id/imageView"
        android:layout_width="15mm"
        android:layout_height="15mm"
        android:src="@drawable/lostnfoundpxnbg"
        android:contentDescription="@string/todo"
        android:layout_marginTop="140dp"/>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/lost_n_found"
        android:gravity="center"
        android:layout_marginTop="20dp"
        android:textStyle="bold"
        android:textSize="20dp"/>

    <ProgressBar
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="30dp"/>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Developed By Jatin Gangare"
        android:gravity="center"
        android:layout_marginTop="200dp"
        />
</LinearLayout>
```

Login Screen :

LoginActivity.java

```
package com.kundevahal.lostnfound;

public class LoginActivity extends AppCompatActivity {
    TextInputEditText editTextEmail, editTextPassword;
    Button ButtonLogin;
    FirebaseAuth mAuth;
    ProgressBar progressBar;
    TextView textView, AuthtextView, forgotPassTextView;
    ImageButton imageButton;

    @Override
    public void onStart() {
        super.onStart();
        // Check if user is signed in (non-null) and update UI accordingly.
        FirebaseUser currentUser = mAuth.getCurrentUser();
        if (currentUser != null && currentUser.isEmailVerified()) {
            checkUserTypeAndRedirect(currentUser.getEmail());
        }
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        mAuth = FirebaseAuth.getInstance();
        editTextEmail = findViewById(R.id.email);
        editTextPassword = findViewById(R.id.pass);
        ButtonLogin = findViewById(R.id.btn_login);
        progressBar = findViewById(R.id.progressbar);
        textView = findViewById(R.id.SigninNow);
        forgotPassTextView = findViewById(R.id.forgotpass);
        imageButton = findViewById(R.id.authsignin);

        // Check for internet connectivity
        if (!isNetworkAvailable()) {
            Toast.makeText(LoginActivity.this, "No internet connection",
Toast.LENGTH_SHORT).show();
        }

        imageButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent Intent = new Intent(getApplicationContext(), AuthenticatorActivity.class);
                startActivity(Intent);
```

```
            finish();
        }
    });

    textView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent Intent = new Intent(getApplicationContext(), SignupActivity.class);
            startActivity(Intent);
            finish();
        }
    });


    ButtonLogin.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            progressBar.setVisibility(View.VISIBLE);
            String email, password;
            email = editTextEmail.getText().toString();
            password = editTextPassword.getText().toString();

            if (TextUtils.isEmpty(email)) {
                Toast.makeText(LoginActivity.this, "Enter Email",
Toast.LENGTH_SHORT).show();
                progressBar.setVisibility(View.GONE);
                return;
            }

            if (TextUtils.isEmpty(password)) {
                Toast.makeText(LoginActivity.this, "Enter Password",
Toast.LENGTH_SHORT).show();
                progressBar.setVisibility(View.GONE);
                return;
            }

            mAuth.signInWithEmailAndPassword(email, password)
                    .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
                        @Override
                        public void onComplete(@NonNull Task<AuthResult> task) {
                            progressBar.setVisibility(View.GONE);
                            if (task.isSuccessful()) {
                                FirebaseUser user = mAuth.getCurrentUser();
                                if (user != null && user.isEmailVerified()) {
                                    checkUserTypeAndRedirect(email);
                                } else {
                                    Toast.makeText(LoginActivity.this, "Please verify your email
address.", Toast.LENGTH_SHORT).show();
                                }
                            } else {
```

```
                        Toast.makeText(LoginActivity.this, "Invalid Email or Password !",
Toast.LENGTH_SHORT).show();
                    }
                }
            });
        }
    });

    forgotPassTextView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // Create an alert dialog to ask for the user's email
            final EditText resetEmail = new EditText(v.getContext());

resetEmail.setInputType(InputType.TYPE_TEXT_VARIATION_EMAIL_ADDRESS);
            resetEmail.setHint("Enter your email");

            AlertDialog.Builder passwordResetDialog = new
AlertDialog.Builder(v.getContext());
            passwordResetDialog.setTitle("Reset Password?");
            passwordResetDialog.setMessage("Enter your email to receive the reset link.");
            passwordResetDialog.setView(resetEmail);

            passwordResetDialog.setPositiveButton("Send", new
DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    // Get the email entered by the user
                    String uemail = resetEmail.getText().toString().trim();
                    if (TextUtils.isEmpty(uemail)) {
                        Toast.makeText(LoginActivity.this, "Please enter your email",
Toast.LENGTH_SHORT).show();
                    } else {

                        // Directly send reset link without checking if the email exists
                        mAuth.sendPasswordResetEmail(uemail).addOnCompleteListener(new
OnCompleteListener<Void>() {
                            @Override
                            public void onComplete(@NonNull Task<Void> task) {
                                if (task.isSuccessful()) {
                                    Toast.makeText(LoginActivity.this, "Reset link sent to your email",
Toast.LENGTH_SHORT).show();
                                } else {
                                    Exception e = task.getException();
                                    if (e != null) {
                                        Log.e("ResetEmail", "Failed to send reset email: " +
e.getMessage());
                                    }
                                    Toast.makeText(LoginActivity.this, "Error! Reset link is not sent",
Toast.LENGTH_SHORT).show();
```

```
                        }}
                    });
                }
            }
        });

        passwordResetDialog.setNegativeButton("Cancel", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                // Close the dialog
                dialog.dismiss();
            }
        });
        passwordResetDialog.create().show();
    }
  });
}

private void checkUserTypeAndRedirect(String email) {
    DatabaseReference databaseRef = FirebaseDatabase.getInstance().getReference();
databaseRef.child("Authenticator").orderByChild("email").equalTo(email).addListenerForSi
ngleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if (dataSnapshot.exists()) {
                startActivity(new Intent(LoginActivity.this, AuthenticatorMainActivity.class));
            } else {
                startActivity(new Intent(LoginActivity.this, UserMainActivity.class));
            }
            finish();
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {
            Toast.makeText(LoginActivity.this, "Database error: " +
databaseError.getMessage(), Toast.LENGTH_SHORT).show();
        }
    });
}

private boolean isNetworkAvailable() {
    ConnectivityManager connectivityManager = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo activeNetworkInfo = connectivityManager.getActiveNetworkInfo();
    return activeNetworkInfo != null && activeNetworkInfo.isConnected();
}
}
```

2. activity_login.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    android:background="@drawable/background_gradient"
    tools:context=".LoginActivity">

    <!-- ImageButton at the top-right corner -->
    <ImageButton
        android:id="@+id/authsignin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/baseline_account_circle_24"
        android:contentDescription="Authenticator Sign in"
        android:background="@android:color/transparent"
        android:padding="10dp"
        android:layout_alignParentTop="true"
        android:layout_alignParentEnd="true" />

    <!-- Center content -->
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:gravity="center"
        android:layout_below="@id/authsignin">

        <ImageView
            android:layout_width="180dp"
            android:layout_height="180dp"
            android:src="@drawable/loginpage"
            android:layout_marginBottom="10dp" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="52dp"
            android:fontFamily="serif"
            android:gravity="center"
            android:text="@string/Login"
            android:textColor="#51A9D1"
            android:textSize="30sp"
            android:textStyle="bold"
            android:layout_marginBottom="24dp" />
```

```
<TextView
        android:id="@+id/forgotpass"
        android:layout_width="match_parent"
        android:layout_height="22dp"
        android:fontFamily="serif"
        android:gravity="right|bottom"
        android:text="Forgot Password ?"
        android:textColor="#51A9D1"
        android:textSize="12sp"
        android:textStyle="bold"
        android:layout_marginRight="15dp"
        android:layout_marginBottom="4dp" />

    <com.google.android.material.textfield.TextInputLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginHorizontal="20dp"
        android:layout_marginBottom="16dp"
        app:boxBackgroundMode="outline"
        app:boxCornerRadiusTopStart="8dp"
        app:boxCornerRadiusTopEnd="8dp"
        app:boxCornerRadiusBottomStart="8dp"
        app:boxCornerRadiusBottomEnd="8dp"
        app:boxStrokeColor="#51A9D1"
        app:hintTextColor="@color/cyan">

        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/email"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="@string/email"
            android:inputType="textEmailAddress"
            android:autofillHints="emailAddress"
            android:textColor="@android:color/black"
            android:textColorHint="@color/black"
            android:importantForAccessibility="yes"
            android:focusable="true"
            android:focusableInTouchMode="true" />
    </com.google.android.material.textfield.TextInputLayout>

    <com.google.android.material.textfield.TextInputLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginHorizontal="20dp"
        app:passwordToggleEnabled="true"
        android:layout_marginBottom="16dp"
        app:boxBackgroundMode="outline"
        app:boxCornerRadiusTopStart="8dp"
        app:boxCornerRadiusTopEnd="8dp"
        app:boxCornerRadiusBottomStart="8dp"
```

```
        app:boxCornerRadiusBottomEnd="8dp"
        app:boxStrokeColor="#51A9D1"
        app:hintTextColor="@color/cyan">

        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/pass"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="@string/Password"
            android:inputType="textPassword"
            android:autofillHints="password"
            android:textColor="@android:color/black"
            android:textColorHint="@color/black"
            android:importantForAccessibility="yes"
            android:focusable="true"
            android:focusableInTouchMode="true" />
    </com.google.android.material.textfield.TextInputLayout>

    <ProgressBar
        android:id="@+id/progressbar"
        android:visibility="gone"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="24dp" />

    <Button
        android:id="@+id/btn_login"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/Login"
        app:cornerRadius="10dp"
        android:backgroundTint="#51A9D1"
        android:textColor="@android:color/white"
        android:layout_marginTop="10dp" />

    <TextView
        android:id="@+id/SigninNow"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/click_to_Sign_in"
        android:textSize="15sp"
        android:gravity="center"
        android:layout_marginTop="20dp"
        android:minHeight="25dp"
        android:padding="8dp" />

  </LinearLayout>
</RelativeLayout>
```

Sign Up :

User Signup :

1. SignupActivity.java

```
package com.kundevahal.lostnfound;

public class SignupActivity extends AppCompatActivity {

   TextInputEditText Firstname, Middlename, Lastname, contactNo, editTextEmail,
editTextPassword;
   Button ButtonSignin;
   FirebaseAuth mAuth;
   ProgressBar progressBar;
   TextView textView;

   FirebaseDatabase db;
   DatabaseReference reference;

   @Override
   public void onStart() {
      super.onStart();
      // Check if user is signed in (non-null) and update UI accordingly.
      FirebaseUser currentUser = mAuth.getCurrentUser();
      if (currentUser != null && currentUser.isEmailVerified()) {
         Intent intent = new Intent(getApplicationContext(), MainActivity.class);
         startActivity(intent);
         finish();
      }
   }

   @SuppressLint("MissingInflatedId")
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_signup);

      mAuth = FirebaseAuth.getInstance();
      editTextEmail = findViewById(R.id.email);
      editTextPassword = findViewById(R.id.pass);
      ButtonSignin = findViewById(R.id.btn_signin);
      progressBar = findViewById(R.id.progressbar);
      textView = findViewById(R.id.LoginNow);
      Firstname = findViewById(R.id.First_name);
      Middlename = findViewById(R.id.middle_name);
      Lastname = findViewById(R.id.last_name);
      contactNo = findViewById(R.id.Mobile_Number);
```

```
        textView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
                startActivity(intent);
                finish();
            }
        });

        ButtonSignin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                progressBar.setVisibility(View.VISIBLE);
                String email, password, fname, Mname, lname, contact;
                email = String.valueOf(editTextEmail.getText().toString().replaceAll("\\s+",""));
                password = String.valueOf(editTextPassword.getText().toString());
                fname = String.valueOf(Firstname.getText().toString().replaceAll("\\s+",""));
                Mname = String.valueOf(Middlename.getText().toString().replaceAll("\\s+",""));
                lname = String.valueOf(Lastname.getText().toString().replaceAll("\\s+",""));
                contact = String.valueOf(contactNo.getText().toString());
                boolean isValid = true;

                if (TextUtils.isEmpty(email)) {
                    editTextEmail.setError("Enter Email");
                    isValid = false;
                }

                if (TextUtils.isEmpty(password)) {
                    Toast.makeText(SignupActivity.this," Password cannot be
empty",Toast.LENGTH_SHORT).show();
                    isValid = false;
                }else if (password.length() < 8) {
                    editTextPassword.setError("Password must be at least 8 characters");
                    isValid = false;
                } else if (!password.matches(".*[!@#$%^&*(),.?\":{}|<>].*")) {
                    editTextPassword.setError("Password must contain at least one special
character");
                    isValid = false;
                } else if (!password.matches(".*[A-Za-z].*")) {
                    editTextPassword.setError("Password must contain at least one letter");
                    isValid = false;
                } else if (!password.matches(".*\\d.*")) {
                    editTextPassword.setError("Password must contain at least one digit");
                    isValid = false;
                }

                if (TextUtils.isEmpty(fname) || !fname.matches("[a-zA-Z ]+")) {
                    Firstname.setError("Enter a valid First Name");
                    isValid = false;
                }
```

```
        if (TextUtils.isEmpty(Mname) || !Mname.matches("[a-zA-Z ]+")) {
            Middlename.setError("Enter a valid Father Name");
            isValid = false;
        }

        if (TextUtils.isEmpty(lname) || !lname.matches("[a-zA-Z ]+")) {
            Lastname.setError("Enter a valid Last Name");
            isValid = false;
        }

        if (TextUtils.isEmpty(contact) || !contact.matches("^[789][0-9]{9}$")) {
            contactNo.setError("Enter a valid Contact (10 digits)");
            isValid = false;
        }

        if (!isValid) {
            progressBar.setVisibility(View.GONE);
            return;
        }

        mAuth.createUserWithEmailAndPassword(email, password)
                .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        progressBar.setVisibility(View.GONE);
                        if (task.isSuccessful()) {

mAuth.getCurrentUser().sendEmailVerification().addOnCompleteListener(new
OnCompleteListener<Void>() {
                                @Override
                                public void onComplete(@NonNull Task<Void> task) {
                                    if (task.isSuccessful()) {

                                        DatabaseReference userref =
FirebaseDatabase.getInstance().getReference("Users");
                                        String UserID = userref.push().getKey();
                                        Users users = new Users(fname, Mname, lname, contact,
email);
                                        userref.child(UserID).setValue(users);

                                        editTextEmail.setText("");
                                        editTextPassword.setText("");
                                        Firstname.setText("");
                                        Middlename.setText("");
                                        Lastname.setText("");
                                        contactNo.setText("");

                                        editTextEmail.clearFocus();
                                        editTextPassword.clearFocus();
```

```
                                    Firstname.clearFocus();
                                    Middlename.clearFocus();
                                    Lastname.clearFocus();
                                    contactNo.clearFocus();

                                    editTextEmail.setError(null);
                                    editTextPassword.setError(null);
                                    Firstname.setError(null);
                                    Middlename.setError(null);
                                    Lastname.setError(null);
                                    contactNo.setError(null);

                                    Toast.makeText(SignupActivity.this, "User Registered
Successfully, please verify your Email",
                                            Toast.LENGTH_SHORT).show();
                                    mAuth.signOut(); // Sign out the user to prevent auto-login
                                    Intent intent = new Intent(getApplicationContext(),
LoginActivity.class);
                                    startActivity(intent);
                                    finish();
                                } else {
                                    Toast.makeText(SignupActivity.this, "Failed To send
Verification Email",
                                            Toast.LENGTH_SHORT).show();
                                }
                            }
                        });

                    } else {
                        // If sign in fails, display a message to the user.
                        Toast.makeText(SignupActivity.this, "Authentication failed.",
                                Toast.LENGTH_SHORT).show();

                    }
                }
            });
        }
    });
  }
}
```

Authenticator Signup :

2. AuthenticatorActivity.java

```
package com.kundevahal.lostnfound;

public class AuthenticatorActivity extends AppCompatActivity {

   TextInputEditText Firstname, Middlename, Lastname, contactNo, editTextEmail,
editTextPassword, specialcode;
   Button ButtonSignin;
   FirebaseAuth mAuth;
   ProgressBar progressBar;
   TextView textView;
   private final String spcode = "Kundevahal@2004";
   String[] locations = {"Grampanchayat Office", "Kuluaai Mandir"};
   AutoCompleteTextView autoCompleteTextView;
   ArrayAdapter<String> arrayAdapter;

   @Override
   public void onStart() {
      super.onStart();
      FirebaseUser currentUser = mAuth.getCurrentUser();
      if (currentUser != null && currentUser.isEmailVerified()) {
         Intent intent = new Intent(getApplicationContext(), MainActivity.class);
         startActivity(intent);
         finish();
      }
   }

   @SuppressLint("MissingInflatedId")
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      requestWindowFeature(Window.FEATURE_NO_TITLE);
      this.getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);
      getSupportActionBar().hide();
      setContentView(R.layout.activity_authenticator);

      mAuth = FirebaseAuth.getInstance();
      editTextEmail = findViewById(R.id.email);
      editTextPassword = findViewById(R.id.pass);
      ButtonSignin = findViewById(R.id.btn_signin);
      progressBar = findViewById(R.id.progressbar);
      textView = findViewById(R.id.LoginNow);
      Firstname = findViewById(R.id.First_name);
      Middlename = findViewById(R.id.middle_name);
      Lastname = findViewById(R.id.last_name);
```

```
        contactNo = findViewById(R.id.Mobile_Number);
        autoCompleteTextView = findViewById(R.id.location);
        arrayAdapter = new ArrayAdapter<>(this, R.layout.list_location, locations);
        autoCompleteTextView.setAdapter(arrayAdapter);
        specialcode = findViewById(R.id.authenticator_edittext);

        autoCompleteTextView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
                String location = parent.getItemAtPosition(position).toString();
                Toast.makeText(AuthenticatorActivity.this, "Location " + location,
Toast.LENGTH_SHORT).show();
            }
        });

        textView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
                startActivity(intent);
                finish();
            }
        });

        ButtonSignin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                progressBar.setVisibility(View.VISIBLE);
                String email, password, fname, Mname, lname, contact, location, Specialcode;
                email = String.valueOf(editTextEmail.getText().toString());
                password = String.valueOf(editTextPassword.getText().toString());
                fname = String.valueOf(Firstname.getText().toString());
                Mname = String.valueOf(Middlename.getText().toString());
                lname = String.valueOf(Lastname.getText().toString());
                contact = String.valueOf(contactNo.getText().toString());
                location = String.valueOf(autoCompleteTextView.getText().toString());
                Specialcode = String.valueOf(specialcode.getText().toString());

                boolean isValid = true;

                if (TextUtils.isEmpty(email)) {
                    editTextEmail.setError("Enter Email");
                    isValid = false;
                } else if (!Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
                    editTextEmail.setError("Enter a valid email");
                    isValid = false;
                }

                if (TextUtils.isEmpty(password)) {
```

```
            Toast.makeText(AuthenticatorActivity.this," Password cannot be
empty",Toast.LENGTH_SHORT).show();
            isValid = false;
        } else if (password.length() < 8) {
            editTextPassword.setError("Password must be at least 6 characters");
            isValid = false;
        } else if (!password.matches(".*[!@#$%^&*(),.?\":{}|<>].*")) {
            editTextPassword.setError("Password must contain at least one special
character");
            isValid = false;
        } else if (!password.matches(".*[A-Za-z].*")) {
            editTextPassword.setError("Password must contain at least one letter");
            isValid = false;
        } else if (!password.matches(".*\\d.*")) {
            editTextPassword.setError("Password must contain at least one digit");
            isValid = false;
        }

        if (TextUtils.isEmpty(fname) || !fname.matches("[a-zA-Z ]+")) {
            Firstname.setError("Enter a valid First Name");
            isValid = false;
        }

        if (TextUtils.isEmpty(Mname) || !Mname.matches("[a-zA-Z ]+")) {
            Middlename.setError("Enter a valid Middle Name");
            isValid = false;
        }

        if (TextUtils.isEmpty(lname) || !lname.matches("[a-zA-Z ]+")) {
            Lastname.setError("Enter a valid Last Name");
            isValid = false;
        }

        if (TextUtils.isEmpty(contact) || !contact.matches("^[1789][0-9]{9}$")) {
            contactNo.setError("Enter a valid 10-digit contact number");
            isValid = false;
        }

        if (TextUtils.isEmpty(location)) {
            autoCompleteTextView.setError("Enter Location");
            isValid = false;
        }

        if (!Specialcode.equals(spcode)) {
            Toast.makeText(AuthenticatorActivity.this,"Invalid Special
Code",Toast.LENGTH_SHORT).show();
            isValid = false;
        }

        if (!isValid) {
```

```
            progressBar.setVisibility(View.GONE);
            return;
        }

        mAuth.createUserWithEmailAndPassword(email, password)
                .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        progressBar.setVisibility(View.GONE);
                        if (task.isSuccessful()) {

mAuth.getCurrentUser().sendEmailVerification().addOnCompleteListener(new
OnCompleteListener<Void>() {
                                @Override
                                public void onComplete(@NonNull Task<Void> task) {
                                    if (task.isSuccessful()) {
                                        DatabaseReference authRef =
FirebaseDatabase.getInstance().getReference("Authenticator");
                                        String AuthID = authRef.push().getKey();
                                        Authenticator auth = new Authenticator(fname, Mname, lname,
contact, location, email);
                                        authRef.child(AuthID).setValue(auth);

                                        Toast.makeText(AuthenticatorActivity.this, "User Registered
Successfully, please verify your Email", Toast.LENGTH_SHORT).show();
                                        mAuth.signOut(); // Sign out the user to prevent auto-login
                                        Intent intent = new Intent(getApplicationContext(),
LoginActivity.class);
                                        startActivity(intent);
                                        finish();
                                    } else {
                                        Toast.makeText(AuthenticatorActivity.this, "Failed To send
Verification Email", Toast.LENGTH_SHORT).show();
                                    }
                                }
                            });
                        } else {
                            Toast.makeText(AuthenticatorActivity.this, "Authentication failed.",
Toast.LENGTH_SHORT).show();
                        }
                    }
                });
        }
    });
  }
}
```

User Side :

1. UserMainActivity.java

```
package com.kundevahal.lostnfound;
public class UserMainActivity extends AppCompatActivity {

    ActivityUserMainBinding binding;
    FirebaseUser user;
    FirebaseAuth auth;
    DatabaseReference databaseReference;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityUserMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        replaceFragment(new HomeFragment());

        databaseReference = FirebaseDatabase.getInstance().getReference("accepted_items");

        listenForNewAcceptedItems();

        // Set up the BottomNavigationView
        binding.bottomNavigationView.setOnItemSelectedListener(item -> {
            int itemId = item.getItemId();
            if (itemId == R.id.user_home) {
                replaceFragment(new HomeFragment());
            }else if (itemId == R.id.user_got_item) {
                replaceFragment(new GotItemFragment());
            } else if (itemId == R.id.user_accepted_request) {
                replaceFragment(new AcceptedRequestFragment());
            }else {
                return false;
            }
            return true;
        });
    }

    private void replaceFragment(Fragment fragment) {
        FragmentManager fragmentManager = getSupportFragmentManager();
        FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
        fragmentTransaction.replace(R.id.frame_layout, fragment);
        fragmentTransaction.commit();
    }
    private void listenForNewAcceptedItems() {
        auth = FirebaseAuth.getInstance();
        user = auth.getCurrentUser();
        final String currentUserEmail = user.getEmail(); // Get the email of the current user
```

```
        databaseReference.addChildEventListener(new ChildEventListener() {
            @Override
            public void onChildAdded(@NonNull DataSnapshot snapshot, String
previousChildName) {
                // A new lost item has been added
                String itemName = snapshot.child("name").getValue(String.class);
                String dateTime = snapshot.child("dateTime").getValue(String.class);

                // Send a notification with the lost item details
                String message = itemName + " on " + dateTime;
                NotificationHelper notificationHelper = new NotificationHelper();

notificationHelper.sendNotificationToUsers(UserMainActivity.this,currentUserEmail, "Item
Accepted ", message);
            }
}
```

2. HomeFragment.java

```
package com.kundevahal.lostnfound;

public class HomeFragment extends Fragment {

    RecyclerView recyclerView;
    DatabaseReference databaseReference,lostItemsRef;
    ArrayList<Item> list;
    ArrayList<Item> originalList;
    ItemAdapter itemAdapter;
    private FloatingActionButton fab;
    private FirebaseAuth mAuth;

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.fragment_home, container, false);

        fab = view.findViewById(R.id.fab);

        // Initialize Firebase Auth
        mAuth = FirebaseAuth.getInstance();
        FirebaseUser currentUser = mAuth.getCurrentUser();

        recyclerView = view.findViewById(R.id.all_item_list);
        databaseReference = FirebaseDatabase.getInstance().getReference("lost_items");
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));

        list = new ArrayList<>();
```

```
        originalList = new ArrayList<>();
        itemAdapter = new ItemAdapter(getContext(), list);
        recyclerView.setAdapter(itemAdapter);


        // Check if there is a logged-in user
        if (currentUser == null) {
            // No user is logged in, redirect to LoginActivity
            redirectToLogin();
        }

        // Set up search functionality
        SearchView searchView = view.findViewById(R.id.user_search_view);
        searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
            @Override
            public boolean onQueryTextSubmit(String query) {
                return false;
            }

            @Override
            public boolean onQueryTextChange(String newText) {
                filter(newText);
                return true;
            }
        });

        loadItems();

        // Set up account circle button click listener
        ImageButton accountCircleButton = view.findViewById(R.id.account_circle_button);
        accountCircleButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Navigate to ProfileFragment
                Fragment profileFragment = new ProfileFragment();
                FragmentTransaction transaction = getFragmentManager().beginTransaction();
                transaction.replace(R.id.frame_layout, profileFragment);
                transaction.addToBackStack(null);
                transaction.commit();
            }
        });

        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Vibrator vibrator = (Vibrator)
requireContext().getSystemService(Context.VIBRATOR_SERVICE);
                if (vibrator != null) {
                    if (android.os.Build.VERSION.SDK_INT >=
android.os.Build.VERSION_CODES.O) {
```

```
                vibrator.vibrate(VibrationEffect.createOneShot(100,
VibrationEffect.DEFAULT_AMPLITUDE));
            } else {
                vibrator.vibrate(100);
            }
        }
        openAddItemFragment();
    }
});

ImageButton notificationButton = view.findViewById(R.id.notification_button);
notificationButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivity(new Intent(getActivity(), UserNotificationActivity.class));
    }
});

return view;
}

private void loadItems() {
    FirebaseUser currentUser = mAuth.getCurrentUser();
    if (currentUser == null) {
        redirectToLogin();
        return;
    }
    final String currentUserEmail = currentUser.getEmail(); // Get the email of the current
user

    databaseReference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            list.clear();
            originalList.clear();
            for (DataSnapshot dataSnapshot : snapshot.getChildren()) {
                Item item = dataSnapshot.getValue(Item.class);
                if (item != null && currentUserEmail != null &&
currentUserEmail.equals(item.getUserEmail())) {
                    // Only add items that belong to the logged-in user
                    list.add(item);
                    originalList.add(item);
                }
            }
            itemAdapter.notifyDataSetChanged();
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {
            Log.e("Firebase Error", error.getMessage());
```

```
            Toast.makeText(getContext(), "Failed to load items.",
Toast.LENGTH_SHORT).show();
        }
    });
  }

  private void filter(String query) {
    ArrayList<Item> filteredList = new ArrayList<>();

    for (Item item : originalList) {
      if (item.getName().toLowerCase().contains(query.toLowerCase())) {
        filteredList.add(item);
      }
    }

    if (filteredList.isEmpty()) {
      Toast.makeText(getContext(), "No item found", Toast.LENGTH_SHORT).show();
    }


    list.clear();
    list.addAll(filteredList);
    itemAdapter.notifyDataSetChanged();
  }

  private void redirectToLogin() {
    Intent intent = new Intent(getActivity(), LoginActivity.class);
    intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
    startActivity(intent);
  }

  private void openAddItemFragment() {
    AddItemFragment addItemFragment = new AddItemFragment();
    FragmentTransaction transaction =
getActivity().getSupportFragmentManager().beginTransaction();
    transaction.replace(R.id.frame_layout, addItemFragment); // Ensure this ID is correct
    transaction.addToBackStack(null);
    transaction.commit();
  }
}
```

3. AddItemFragment.java

```
package com.kundevahal.lostnfound;

public class AddItemFragment extends Fragment {
```

```
    private static final int REQUEST_IMAGE_PICK = 1;
    private static final int REQUEST_ITEM_IMAGE_PICK = 5;
    private static final int REQUEST_ADHAR_IMAGE_PICK = 4;
    private static final int REQUEST_ADHAR_IMAGE_PICK_CAMERA = 6;
    private static final int REQUEST_STORAGE_PERMISSION = 2;

    private ImageView itemPhoto, adharPhoto;
    private EditText itemName, itemLocation, adharNumber, itemDescription;
    private TextView itemDateTime,userName, phoneNumber;
    private Button submitButton, cancelButton;

    private Uri imageUri, adharImageUri,photoUri;

    private FirebaseAuth auth;
    private DatabaseReference databaseReference,userdatbaseref;
    private StorageReference storageReference;

    private ProgressDialog progressDialog;
    private FirebaseUser user;
    private Bitmap capturedBitmap;
    private Bitmap capturedBitmapadhar;
    private ChipGroup chipGroupImageType;
    private Chip chipOriginalImage, chipLookAlike;
    String firstNameStr,middleNameStr,lastNameStr,mobileNumberStr,fullname;


    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_add_item, container, false);

        auth = FirebaseAuth.getInstance();
        user = auth.getCurrentUser();
        databaseReference = FirebaseDatabase.getInstance().getReference("lost_items");
        storageReference = FirebaseStorage.getInstance().getReference("item_images");
        userdatbaseref = FirebaseDatabase.getInstance().getReference("Users");

        itemPhoto = view.findViewById(R.id.item_photo);
        adharPhoto = view.findViewById(R.id.adhar_card_photo);
        itemName = view.findViewById(R.id.item_name);
        itemLocation = view.findViewById(R.id.item_location);
        adharNumber = view.findViewById(R.id.adhar_number);
        itemDescription = view.findViewById(R.id.item_description);
        userName = view.findViewById(R.id.user_name);
        phoneNumber = view.findViewById(R.id.phone_number);
        itemDateTime = view.findViewById(R.id.item_datetime);
        submitButton = view.findViewById(R.id.submit_button);
        cancelButton = view.findViewById(R.id.cancel_button);
        chipGroupImageType = view.findViewById(R.id.chipGroupImageType);
```

```
        chipOriginalImage = view.findViewById(R.id.chipOriginalImage);
        chipLookAlike = view.findViewById(R.id.chipLookAlike);

        NotificationHelper.createNotificationChannel(getActivity());

        // Set user email
        if (user != null) {
            String email = user.getEmail();

            // Fetch additional user details from the database using email as key
            Query userQuery = userdatbaseref.orderByChild("email").equalTo(email);
            userQuery.addListenerForSingleValueEvent(new ValueEventListener() {
                @Override
                public void onDataChange(@NonNull DataSnapshot snapshot) {
                    if (snapshot.exists()) {
                        for (DataSnapshot userSnapshot : snapshot.getChildren()) {
                            firstNameStr = userSnapshot.child("firstname").getValue(String.class);
                            middleNameStr =
userSnapshot.child("middlename").getValue(String.class);
                            lastNameStr = userSnapshot.child("lastname").getValue(String.class);
                            mobileNumberStr = userSnapshot.child("contact").getValue(String.class);
                            fullname = firstNameStr+" "+middleNameStr+" "+lastNameStr;

                            // Only set userName when data is fully retrieved
                            userName.setText(fullname);
                            phoneNumber.setText(mobileNumberStr);


                        }
                    } else {
                        Log.e(TAG, "No user data found for email: " + email);
                        Toast.makeText(getContext(), "No user data found",
Toast.LENGTH_SHORT).show();
                    }
                }

                @Override
                public void onCancelled(@NonNull DatabaseError error) {
                    Log.e(TAG, "Database error: ", error.toException());
                    Toast.makeText(getContext(), "Failed to retrieve user data",
Toast.LENGTH_SHORT).show();
                }
            });

        }

        // Set a listener for chip selection changes
        chipGroupImageType.setOnCheckedChangeListener((group, checkedId) -> {
            if (checkedId == chipOriginalImage.getId()) {
                Toast.makeText(getContext(), "Original Item Selected",
```

```
Toast.LENGTH_SHORT).show();
            // Handle selection of "Original Item"
        } else if (checkedId == chipLookAlike.getId()) {
            Toast.makeText(getContext(), "Look Alike Selected",
Toast.LENGTH_SHORT).show();
            // Handle selection of "Look Alike"
        }
    });

    itemPhoto.setOnClickListener(v -> showImageSelectionDialog());
    adharPhoto.setOnClickListener(v -> showAdharImageSelectionDialog());
    itemDateTime.setOnClickListener(v -> showDateTimePicker());
    submitButton.setOnClickListener(v -> submitItem());
    cancelButton.setOnClickListener(v -> {
        Fragment homeFragment = new HomeFragment();
        FragmentManager fragmentManager = getParentFragmentManager();
        FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
        fragmentTransaction.replace(R.id.frame_layout, homeFragment);
        fragmentTransaction.addToBackStack(null);
        fragmentTransaction.commit();
    });
    return view;
}

private void showImageSelectionDialog() {
    String[] options = {"Camera", "Gallery"};
    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
    builder.setTitle("Select Item Photo")
            .setItems(options, (dialog, which) -> {
                if (which == 0) {
                    if (ContextCompat.checkSelfPermission(getActivity(),
Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED) {
                        ActivityCompat.requestPermissions(getActivity(), new
String[]{Manifest.permission.CAMERA}, REQUEST_IMAGE_PICK);
                    } else {
                        openCamera();
                    }
                } else {
                    openGallery();
                }
            }).show();
}

private void showAdharImageSelectionDialog() {
    String[] options = {"Camera", "Gallery"};
    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
    builder.setTitle("Select Aadhar Photo")
            .setItems(options, (dialog, which) -> {
                if (which == 0) {
                    if (ContextCompat.checkSelfPermission(getActivity(),
```

```
Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED) {
                    ActivityCompat.requestPermissions(getActivity(), new
String[]{Manifest.permission.CAMERA}, REQUEST_ADHAR_IMAGE_PICK);
                } else {
                    openAdharCamera();
                }
            } else {
                openAdharGallery();
            }
        }).show();
    }

    private void openCamera() {
        Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        if(cameraIntent.resolveActivity(getActivity().getPackageManager())!=null)
        {
            startActivityForResult(cameraIntent, REQUEST_ITEM_IMAGE_PICK);
        }

    }

    private void openGallery() {
        Intent galleryIntent = new Intent(Intent.ACTION_PICK,
MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
        startActivityForResult(galleryIntent, REQUEST_IMAGE_PICK);
    }

    private void openAdharCamera() {
        Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        startActivityForResult(cameraIntent, REQUEST_ADHAR_IMAGE_PICK_CAMERA);
    }

    private void openAdharGallery() {
        Intent galleryIntent = new Intent(Intent.ACTION_PICK,
MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
        startActivityForResult(galleryIntent, REQUEST_ADHAR_IMAGE_PICK);
    }

    private void showDateTimePicker() {
        final Calendar calendar = Calendar.getInstance();
        final Calendar maxDateTime = Calendar.getInstance(); // Current date and time for
validation

        // DatePickerDialog to select the date
        DatePickerDialog datePickerDialog = new DatePickerDialog(getActivity(),
                (view, year, month, dayOfMonth) -> {
                    calendar.set(Calendar.YEAR, year);
                    calendar.set(Calendar.MONTH, month);
                    calendar.set(Calendar.DAY_OF_MONTH, dayOfMonth);
```

```
                // TimePickerDialog to select the time
                TimePickerDialog timePickerDialog = new TimePickerDialog(getActivity(),
                        (view1, hourOfDay, minute) -> {
                            calendar.set(Calendar.HOUR_OF_DAY, hourOfDay);
                            calendar.set(Calendar.MINUTE, minute);

                            // Validate the selected date and time
                            if (calendar.after(maxDateTime)) {
                                Toast.makeText(getContext(), "Future date and time are not allowed.",
Toast.LENGTH_SHORT).show();
                                itemDateTime.setText(""); // Reset the field
                            } else {
                                SimpleDateFormat dateFormat = new
SimpleDateFormat("dd/MM/yyyy HH:mm", Locale.getDefault());
                                itemDateTime.setText(dateFormat.format(calendar.getTime()));
                            }
                        },
                        calendar.get(Calendar.HOUR_OF_DAY),
                        calendar.get(Calendar.MINUTE),
                        false);
                timePickerDialog.show();
            },
            calendar.get(Calendar.YEAR),
            calendar.get(Calendar.MONTH),
            calendar.get(Calendar.DAY_OF_MONTH));

        // Restrict DatePicker to not allow future dates
        datePickerDialog.getDatePicker().setMaxDate(maxDateTime.getTimeInMillis());
        datePickerDialog.show();
    }

    private void submitItem() {
        String name = itemName.getText().toString().trim();
        String location = itemLocation.getText().toString().trim();
        String adhar = adharNumber.getText().toString().trim();
        String description = itemDescription.getText().toString().trim();
        String user = userName.getText().toString().trim();
        String phone = phoneNumber.getText().toString().trim();
        String dateTime = itemDateTime.getText().toString().trim();
        String userEmail = auth.getCurrentUser() != null ? auth.getCurrentUser().getEmail() :
""; // Get the current user email
        String imageType;
        if (chipGroupImageType.getCheckedChipId() == chipOriginalImage.getId()) {
            imageType = "Original Item";
        } else if (chipGroupImageType.getCheckedChipId() == chipLookAlike.getId()) {
            imageType = "Look Alike";
        } else {
            Toast.makeText(getActivity(), "Please select the image type",
Toast.LENGTH_SHORT).show();
            return;
```

```
        }
        String finalImageType = imageType;

        if (name.isEmpty() || location.isEmpty() || adhar.isEmpty() || description.isEmpty() ||
dateTime.isEmpty() || imageUri==null || adharImageUri==null){
            Toast.makeText(getActivity(), "Please fill all fields",
Toast.LENGTH_SHORT).show();
            return;
        }
        else if(adhar.length()!=12)
        {
            Toast.makeText(getActivity(), "Invalid Adhar Number",
Toast.LENGTH_SHORT).show();
            return;
        }
        else if(itemDescription.length()>60)
        {
            Toast.makeText(getActivity(), "Discription overloaded",
Toast.LENGTH_SHORT).show();
            return;
        }
        else if(itemName.length()>30)
        {
            Toast.makeText(getActivity(), "Item Name overloaded",
Toast.LENGTH_SHORT).show();
            return;
        }
        else if(itemLocation.length()>60)
        {
            Toast.makeText(getActivity(), "Location overloaded",
Toast.LENGTH_SHORT).show();
            return;
        }

        progressDialog = new ProgressDialog(getActivity());
        progressDialog.setMessage("Submitting...");
        progressDialog.show();

        // Upload item image
        if (imageUri != null) {
            StorageReference fileReference =
storageReference.child("item_images").child(System.currentTimeMillis() + ".jpg");
            fileReference.putFile(imageUri)
                    .addOnSuccessListener(taskSnapshot ->
fileReference.getDownloadUrl().addOnSuccessListener(uri -> {
                        String imageUrl = uri.toString();
                        // Upload Aadhar image
                        if (adharImageUri != null) {
                            StorageReference adharFileReference =
storageReference.child("adhar_images").child(System.currentTimeMillis() + ".jpg");
```

```
                    adharFileReference.putFile(adharImageUri)
                            .addOnSuccessListener(taskSnapshot1 ->
adharFileReference.getDownloadUrl().addOnSuccessListener(adharUri -> {
                                String adharImageUrl = adharUri.toString();
                                saveItemDetails(name, location, adhar, description, user, phone,
dateTime, imageUrl, adharImageUrl, userEmail,finalImageType);
                            }))
                            .addOnFailureListener(e -> {
                                progressDialog.dismiss();
                                Toast.makeText(getActivity(), "Failed to upload Aadhar image: " +
e.getMessage(), Toast.LENGTH_SHORT).show();
                            });
                } else {
                    //Toast.makeText(getActivity(), "Adhaar Image is Empty",
Toast.LENGTH_SHORT).show();
                    saveItemDetails(name, location, adhar, description, user, phone, dateTime,
imageUrl, null, userEmail,finalImageType);
                }
            }))
            .addOnFailureListener(e -> {
                progressDialog.dismiss();
                Toast.makeText(getActivity(), "Failed to upload item image: " +
e.getMessage(), Toast.LENGTH_SHORT).show();
            });
    } else {
        //Toast.makeText(getActivity(), "Item Image is Empty",
Toast.LENGTH_SHORT).show();
        saveItemDetails(name, location, adhar, description, user, phone, dateTime, null, null,
userEmail,finalImageType);
    }
  }

  private void saveItemDetails(String name, String location, String adhar, String description,
String user, String phone, String dateTime, String imageUrl, String adharImageUrl, String
userEmail, String finalImageType) {
    String itemId = databaseReference.push().getKey();
    Item item = new Item(itemId, name, location, dateTime, imageUrl, adharImageUrl,
description, adhar, user, phone, userEmail,finalImageType);
    databaseReference.child(itemId).setValue(item).addOnCompleteListener(task -> {
        progressDialog.dismiss();
        if (task.isSuccessful()) {
            Toast.makeText(getActivity(), "Item submitted successfully",
Toast.LENGTH_SHORT).show();
            //listenForNewLostItems();

            Fragment homeFragment = new HomeFragment();
            FragmentManager fragmentManager = getParentFragmentManager();
            FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
            fragmentTransaction.replace(R.id.frame_layout, homeFragment);
            fragmentTransaction.addToBackStack(null);
```

```
            fragmentTransaction.commit();

        } else {
            Toast.makeText(getActivity(), "Failed to submit item",
Toast.LENGTH_SHORT).show();
        }
      });
   }

   @Override
   public void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
      super.onActivityResult(requestCode, resultCode, data);

      if (resultCode == Activity.RESULT_OK) {
         if (requestCode == REQUEST_ITEM_IMAGE_PICK) {


            if (data != null && data.getExtras() != null) {
               Bitmap itmbitmap = (Bitmap) data.getExtras().get("data");
               itemPhoto.setImageBitmap(itmbitmap);
               imageUri = getImageUri(getContext(),itmbitmap);
               //capturedBitmap = bitmap;// Save bitmap for further use
            }

        } else if (requestCode == REQUEST_ADHAR_IMAGE_PICK_CAMERA) {
            if (data != null && data.getExtras() != null) {
               Bitmap adhrbitmap = (Bitmap) data.getExtras().get("data");
               adharPhoto.setImageBitmap(adhrbitmap);
               adharImageUri = getImageUri(getContext(),adhrbitmap);
              // capturedBitmapadhar = bitmap;// Save bitmap for further use
            }
        }
      } else {
         Toast.makeText(getContext(), "Photo capture canceled",
Toast.LENGTH_SHORT).show();
      }
   }
   public static Uri getImageUri(Context context, Bitmap bitmap) {
      ByteArrayOutputStream bytes = new ByteArrayOutputStream();
      bitmap.compress(Bitmap.CompressFormat.JPEG, 100, bytes);
      String path = MediaStore.Images.Media.insertImage(context.getContentResolver(),
bitmap, "Title", null);
      return Uri.parse(path);
   }
}
```

4. AcceptedRequestFragment.java

```
package com.kundevahal.lostnfound;

public class AcceptedRequestFragment extends Fragment {

    private RecyclerView recyclerView;
    private UserAcceptedItemsAdapter acceptedItemsAdapter;
    private List<AcceptedItem> acceptedItems = new ArrayList<>();
    private List<AcceptedItem> originalList = new ArrayList<>();
    private FirebaseAuth mAuth;
    private String currentUserEmail;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_accepted_request, container, false);
        recyclerView = view.findViewById(R.id.recyclerViewAcceptedItems);
        recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
        acceptedItemsAdapter = new UserAcceptedItemsAdapter(acceptedItems, getContext());
        recyclerView.setAdapter(acceptedItemsAdapter);

        mAuth = FirebaseAuth.getInstance(); // Initialize Firebase Auth
        loadItems(); // Load accepted items for the logged-in user

        return view;
    }

    private void loadItems() {
        FirebaseUser currentUser = mAuth.getCurrentUser();
        if (currentUser == null) {
            redirectToLogin(); // Redirect to login if no user is logged in
            return;
        }

        currentUserEmail = currentUser.getEmail(); // Get the email of the current user

        DatabaseReference databaseReference =
FirebaseDatabase.getInstance().getReference("accepted_items");
        databaseReference.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                acceptedItems.clear();
                originalList.clear(); // Assuming you want to keep an original list of items

                for (DataSnapshot dataSnapshot : snapshot.getChildren()) {
                    AcceptedItem item = dataSnapshot.getValue(AcceptedItem.class);
                    if (item != null && currentUserEmail != null &&
currentUserEmail.equals(item.getUserEmail())) {
                        // Only add items that belong to the logged-in user
```

```
                acceptedItems.add(item);
                originalList.add(item); // Maintain the original list if needed
            }
        }
        acceptedItemsAdapter.notifyDataSetChanged(); // Notify adapter about data change
    }

    @Override
    public void onCancelled(@NonNull DatabaseError error) {
        Log.e("Firebase Error", error.getMessage());
        Toast.makeText(getContext(), "Failed to load accepted items.",
Toast.LENGTH_SHORT).show();
    }
  });
 }

 private void redirectToLogin() {
  // Your login redirection logic here
  Intent intent = new Intent(getActivity(), LoginActivity.class);
  startActivity(intent);
 }
}
```

5. GotItemFragment.java

```
package com.kundevahal.lostnfound;

public class GotItemFragment extends Fragment {

 private RecyclerView recyclerView;
 private GotItemAdapter gotItemAdapter;
 private List<HandoverItem> gotItems = new ArrayList<>();
 private FirebaseAuth mAuth;
 private String currentUserEmail;

 @Override
 public View onCreateView(LayoutInflater inflater, ViewGroup container,
                          Bundle savedInstanceState) {
  View view = inflater.inflate(R.layout.fragment_got_item, container, false);
  recyclerView = view.findViewById(R.id.recyclerViewGotItems);
  recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
  gotItemAdapter = new GotItemAdapter(gotItems, getContext());
  recyclerView.setAdapter(gotItemAdapter);
  mAuth = FirebaseAuth.getInstance();

  loadItems(); // Load items when the view is created

  return view;
```

```
    }

    private void loadItems() {
        FirebaseUser currentUser = mAuth.getCurrentUser();
        if (currentUser == null) {
            redirectToLogin(); // Redirect to login if no user is logged in
            return;
        }

        currentUserEmail = currentUser.getEmail(); // Get the email of the current user

        DatabaseReference databaseReference =
FirebaseDatabase.getInstance().getReference("handover_records");
        databaseReference.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                gotItems.clear();
                for (DataSnapshot dataSnapshot : snapshot.getChildren()) {
                    HandoverItem item = dataSnapshot.getValue(HandoverItem.class);
                    if (item != null && currentUserEmail != null &&
currentUserEmail.equals(item.getEmail())) {
                        // Only add items that belong to the logged-in user
                        gotItems.add(item);
                    }
                }
                gotItemAdapter.notifyDataSetChanged(); // Notify adapter about data change
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                Log.e("Firebase Error", error.getMessage());
                Toast.makeText(getContext(), "Failed to load handed-over items.",
Toast.LENGTH_SHORT).show();
            }
        });
    }

    private void redirectToLogin() {
        Intent intent = new Intent(getContext(), LoginActivity.class);
        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK); // Clear the back stack
        startActivity(intent);
        getActivity().finish(); // Optionally finish the current activity
    }
}
```

6. ProfileFragment.java

```
package com.kundevahal.lostnfound;

public class ProfileFragment extends Fragment implements OnMapReadyCallback {

    private TextView userEmail, firstName, middleName, lastName, mobileNumber;
    private Button logoutButton, editProfile, deleteButton,redirectmap;
    private FirebaseAuth mAuth;
    private ImageButton backButton;
    private DatabaseReference databaseReference;
    private FirebaseUser user;
    private GoogleMap googleMap;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.fragment_profile, container, false);

        // Initialize Firebase Auth and Database
        mAuth = FirebaseAuth.getInstance();
        databaseReference = FirebaseDatabase.getInstance().getReference("Users");
        user = mAuth.getCurrentUser();

        // Initialize UI elements
        userEmail = view.findViewById(R.id.user_email);
        firstName = view.findViewById(R.id.first_name);
        middleName = view.findViewById(R.id.middle_name);
        lastName = view.findViewById(R.id.last_name);
        mobileNumber = view.findViewById(R.id.mobile_number);
        logoutButton = view.findViewById(R.id.btn_logout);
        backButton = view.findViewById(R.id.back_button);
        editProfile = view.findViewById(R.id.btn_edit_profile);
        deleteButton = view.findViewById(R.id.btn_delete_account);
        redirectmap = view.findViewById(R.id.btn_redirect_map);

        SupportMapFragment supportMapFragment = (SupportMapFragment)
getChildFragmentManager().findFragmentById(R.id.gmap);
        supportMapFragment.getMapAsync(this);

        // Set user email
        if (user != null) {
            String email = user.getEmail();
            userEmail.setText(email);

            // Fetch additional user details from the database using email as key
            Query userQuery = databaseReference.orderByChild("email").equalTo(email);
            userQuery.addListenerForSingleValueEvent(new ValueEventListener() {
                @Override
```

```
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                if (snapshot.exists()) {
                    for (DataSnapshot userSnapshot : snapshot.getChildren()) {
                        String firstNameStr =
userSnapshot.child("firstname").getValue(String.class);
                        String middleNameStr =
userSnapshot.child("middlename").getValue(String.class);
                        String lastNameStr =
userSnapshot.child("lastname").getValue(String.class);
                        String mobileNumberStr =
userSnapshot.child("contact").getValue(String.class);

                    }
                } else {
                    Log.e(TAG, "No user data found for email: " + email);
                    Toast.makeText(getContext(), "No user data found",
Toast.LENGTH_SHORT).show();
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                Log.e(TAG, "Database error: ", error.toException());
                Toast.makeText(getContext(), "Failed to retrieve user data",
Toast.LENGTH_SHORT).show();
            }
        });

        deleteButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                promptUserForPassword(email);
            }
        });
    }

    redirectmap.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // Specify the URL you want to open
            String url = "https://maps.app.goo.gl/VzpfUizzkDMt98kj8";

            // Create an intent to open the URL
            Intent intent = new Intent(Intent.ACTION_VIEW);
            intent.setData(Uri.parse(url));

            startActivity(intent);

        }
    });
```

```
        // Set onClickListener for the logout button
        logoutButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mAuth.signOut();
                Intent intent = new Intent(getActivity(), LoginActivity.class);
                startActivity(intent);
                requireActivity().finish();
            }
        });

        editProfile.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Fragment editProfileFragment = new EditUserProfileFragment();
                FragmentManager fragmentManager = getParentFragmentManager();
                FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
                fragmentTransaction.replace(R.id.frame_layout, editProfileFragment);
                fragmentTransaction.addToBackStack(null);
                fragmentTransaction.commit();
            }
        });

        // Set onClickListener for the back button
        backButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                requireActivity().onBackPressed();
            }
        });

        return view;
    }

    private void promptUserForPassword(String email) {
        AlertDialog.Builder builder = new AlertDialog.Builder(requireContext());
        builder.setTitle("Reauthenticate");
        builder.setMessage("Please enter your password to delete your account");

        // Use EditText to allow user to input password
        final EditText input = new EditText(requireContext());
        input.setInputType(InputType.TYPE_CLASS_TEXT |
InputType.TYPE_TEXT_VARIATION_PASSWORD);
        builder.setView(input);

        builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                String password = input.getText().toString();
```

```
            if (!password.isEmpty()) {
                reauthenticateAndDeleteUserAccount(email, password);
            } else {
                Toast.makeText(getContext(), "Password cannot be empty",
Toast.LENGTH_SHORT).show();
            }
        }
    });

    builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.cancel();
        }
    });

    builder.show();
}

private void reauthenticateAndDeleteUserAccount(String email, String password) {
    AuthCredential credential = EmailAuthProvider.getCredential(email, password);
    user.reauthenticate(credential).addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            Log.d(TAG, "Reauthentication successful.");
            deleteUserAccount(email);
        } else {
            Log.e(TAG, "Reauthentication failed: ", task.getException());
            Toast.makeText(getContext(), "Re-authentication failed: " +
task.getException().getMessage(), Toast.LENGTH_SHORT).show();
        }
    });
}

private void deleteUserAccount(String email) {
    Query userQuery = FirebaseDatabase.getInstance().getReference().child("Users")
            .orderByChild("email").equalTo(email);

    userQuery.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot snapshot) {
            if (snapshot.exists()) {
                Log.d(TAG, "User data found. Proceeding to delete user data.");
                for (DataSnapshot userSnapshot : snapshot.getChildren()) {
                    userSnapshot.getRef().removeValue().addOnCompleteListener(removeTask -
> {
                        if (removeTask.isSuccessful()) {
                            Log.d(TAG, "User data deleted successfully. Proceeding to delete
Firebase Authentication account.");
                            user.delete().addOnCompleteListener(task -> {
                                if (task.isSuccessful()) {
```

```
                            Log.d(TAG, "User account deleted from Firebase Authentication.");
                            Toast.makeText(getContext(), "Account deleted",
Toast.LENGTH_SHORT).show();
                            // Redirect to LoginActivity
                            Intent intent = new Intent(getActivity(), LoginActivity.class);
                            startActivity(intent);
                            requireActivity().finish();
                        } else {
                            Log.e(TAG, "Failed to delete Firebase Authentication account: ",
task.getException());
                            Toast.makeText(getContext(), "Failed to delete account: " +
task.getException().getMessage(), Toast.LENGTH_SHORT).show();
                        }
                    });
                } else {
                    Log.e(TAG, "Failed to delete user data: ", removeTask.getException());
                    Toast.makeText(getContext(), "Failed to delete account data",
Toast.LENGTH_SHORT).show();
                }
            });
        }
    } else {
        Log.e(TAG, "No user data found for email: " + email);
        Toast.makeText(getContext(), "No user data found",
Toast.LENGTH_SHORT).show();
    }
}

@Override
public void onCancelled(DatabaseError error) {
    Log.e(TAG, "Database error: ", error.toException());
    Toast.makeText(getContext(), "Failed to delete account data",
Toast.LENGTH_SHORT).show();
}
});
}

@Override
public void onMapReady(@NonNull GoogleMap googleMap) {
    this.googleMap = googleMap;
    LatLng location= new LatLng(18.969205990358127, 73.06330746871606);
    googleMap.addMarker(new MarkerOptions().position(location).title("Grampanchayat
Kundevahal"));
    googleMap.moveCamera(CameraUpdateFactory.newLatLngZoom(location,18));
}
}
```

7. EditUserProfileFragment.java

```
package com.kundevahal.lostnfound;

public class EditUserProfileFragment extends Fragment {

    private EditText editFirstName, editMiddleName, editLastName, editMobileNumber;
    private Button saveChangesButton, cancelButton;
    private FirebaseAuth mAuth;
    private DatabaseReference databaseReference;
    private String email;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.fragment_edit_user_profile, container, false);

        // Initialize Firebase Auth and Database
        mAuth = FirebaseAuth.getInstance();
        databaseReference = FirebaseDatabase.getInstance().getReference("Users");

        // Initialize UI elements
        editFirstName = view.findViewById(R.id.edit_first_name);
        editMiddleName = view.findViewById(R.id.edit_middle_name);
        editLastName = view.findViewById(R.id.edit_last_name);
        editMobileNumber = view.findViewById(R.id.edit_mobile_number);
        saveChangesButton = view.findViewById(R.id.btn_save_changes);
        cancelButton = view.findViewById(R.id.btn_cancel);

        // Get current user email
        FirebaseUser currentUser = mAuth.getCurrentUser();
        if (currentUser != null) {
            email = currentUser.getEmail();
            loadUserData();
        }

        // Set onClickListener for the Save Changes button
        saveChangesButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                saveChanges();
            }
        });
        cancelButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                cancelEdit();
            }
        });
```

```
        return view;
    }

    private void loadUserData() {
        if (email != null) {
            Query userQuery = databaseReference.orderByChild("email").equalTo(email);
            userQuery.addListenerForSingleValueEvent(new ValueEventListener() {
                @Override
                public void onDataChange(@NonNull DataSnapshot snapshot) {
                    if (snapshot.exists()) {
                        for (DataSnapshot userSnapshot : snapshot.getChildren()) {
                            String firstName = userSnapshot.child("firstname").getValue(String.class);
                            String middleName =
userSnapshot.child("middlename").getValue(String.class);
                            String lastName = userSnapshot.child("lastname").getValue(String.class);
                            String mobileNumber =
userSnapshot.child("contact").getValue(String.class);

                            if (firstName != null) editFirstName.setText(firstName);
                            if (middleName != null) editMiddleName.setText(middleName);
                            if (lastName != null) editLastName.setText(lastName);
                            if (mobileNumber != null) editMobileNumber.setText(mobileNumber);
                        }
                    } else {
                        Toast.makeText(getActivity(), "User data not found",
Toast.LENGTH_SHORT).show();
                    }
                }
            });
        }
    }

    private void saveChanges() {
        String firstName = editFirstName.getText().toString().trim();
        String middleName = editMiddleName.getText().toString().trim();
        String lastName = editLastName.getText().toString().trim();
        String mobileNumber = editMobileNumber.getText().toString().trim();

        boolean isValid = true;

        if (TextUtils.isEmpty(firstName) || !firstName.matches("[a-zA-Z ]+")) {
            editFirstName.setError("Enter a valid First Name");
            isValid = false;
        }

        if (TextUtils.isEmpty(middleName) || !middleName.matches("[a-zA-Z ]+")) {
            editMiddleName.setError("Enter a valid Middle Name");
            isValid = false;
        }
```

```
        if (TextUtils.isEmpty(lastName) || !lastName.matches("[a-zA-Z ]+")) {
            editLastName.setError("Enter a valid Last Name");
            isValid = false;
        }

        if (TextUtils.isEmpty(mobileNumber) || mobileNumber.length() != 10 ||
!mobileNumber.matches("[0-9]+")) {
            editMobileNumber.setError("Enter a valid Mobile Number (10 digits)");
            isValid = false;
        }

        if (!isValid) {
            return;
        }

        if (email != null) {
            Query userQuery = databaseReference.orderByChild("email").equalTo(email);
            userQuery.addListenerForSingleValueEvent(new ValueEventListener() {
                @Override
                public void onDataChange(@NonNull DataSnapshot snapshot) {
                    if (snapshot.exists()) {
                        for (DataSnapshot userSnapshot : snapshot.getChildren()) {
                            userSnapshot.getRef().child("firstname").setValue(firstName);
                            userSnapshot.getRef().child("middlename").setValue(middleName);
                            userSnapshot.getRef().child("lastname").setValue(lastName);
                            userSnapshot.getRef().child("contact").setValue(mobileNumber)
                                    .addOnCompleteListener(new OnCompleteListener<Void>() {
                                        @Override
                                        public void onComplete(@NonNull Task<Void> task) {
                                            if (task.isSuccessful()) {
                                                Toast.makeText(getActivity(), "Profile updated",
Toast.LENGTH_SHORT).show();
                                                getFragmentManager().popBackStack();
                                            } else {
                                                Toast.makeText(getActivity(), "Failed to update profile",
Toast.LENGTH_SHORT).show();
                                            }
                                        }
                                    });
                        }
                    }
                }
            });
        }
    }
    private void cancelEdit() {
        if (getFragmentManager() != null) {
            getFragmentManager().popBackStack();
        }
    }
}
```

Authenticator Side :

1. AuthenticatorMainActivity.java

```
package com.kundevahal.lostnfound;

public class AuthenticatorMainActivity extends AppCompatActivity {

    FirebaseAuth auth;
    FirebaseUser user;
    private RelativeLayout mainLayout;

    RecyclerView recyclerView;
    DatabaseReference databaseReference;
    ItemAdapter itemAdapter;
    ArrayList<Item> list;
    ArrayList<Item> originalList;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_authenticator_main);

        auth = FirebaseAuth.getInstance();
        MaterialToolbar toolbar = findViewById(R.id.topAppbar);
        DrawerLayout drawerLayout = findViewById(R.id.drawer_layout);
        NavigationView navigationView = findViewById(R.id.navigation_view);
        mainLayout = findViewById(R.id.main_layout);
        View headerView = navigationView.getHeaderView(0);
        TextView headerEmail = headerView.findViewById(R.id.email);
        recyclerView = findViewById(R.id.itemlist);
        databaseReference = FirebaseDatabase.getInstance().getReference("lost_items");
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));

        list = new ArrayList<>();
        originalList = new ArrayList<>();
        itemAdapter = new ItemAdapter(this, list);
        recyclerView.setAdapter(itemAdapter);

        NotificationHelper notificationHelper = new NotificationHelper();
        listenForNewLostItems();
        //notificationHelper.listenForNewItemAdded(this);


        toolbar.setNavigationOnClickListener(v ->
drawerLayout.openDrawer(GravityCompat.START));

        user = auth.getCurrentUser();
        if (user == null) {
```

```
            Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
            startActivity(intent);
            finish();
        } else {
            headerEmail.setText(user.getEmail());
            loadAllItems();
        }

        navigationView.setNavigationItemSelectedListener(item -> {
            int id = item.getItemId();
            drawerLayout.closeDrawer(GravityCompat.START);
            if (id == R.id.nav_account) {
                startActivity(new Intent(getApplicationContext(), AccountActivity.class));
                finish();
            } else if (id == R.id.nav_home) {
                // Stay on the current activity
            } else if (id == R.id.nav_acceptreq) {
                goToAcceptedItemFragment();
            } else if (id == R.id.nav_item_handed_over) {
                hideViews();
                AuthItemHandoverFragment authItemHandoverFragment = new
AuthItemHandoverFragment();
                getSupportFragmentManager().beginTransaction()
                        .replace(R.id.fragment_container, authItemHandoverFragment)
                        .addToBackStack(null)
                        .commit();
                showToast("Item Handed Over is clicked");
            } else if (id == R.id.nav_logout) {
                logoutUser();
            }
            return true;
        });

        // Search functionality
        SearchView searchView = findViewById(R.id.search_view);
        searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
            @Override
            public boolean onQueryTextSubmit(String query) {
                return false;
            }

            @Override
            public boolean onQueryTextChange(String newText) {
                ArrayList<Item> filteredList = filter(newText);
                if (filteredList.isEmpty()) {
                    Toast.makeText(AuthenticatorMainActivity.this, "No item found",
Toast.LENGTH_SHORT).show();
                }
                return true;
            }
```

```
        });

        ImageButton NotificationButton = findViewById(R.id.notification_button_auth);
        NotificationButton.setOnClickListener(v -> startActivity(new
Intent(AuthenticatorMainActivity.this, AuthenticatorNotificationActivity.class)));
    }

    private void loadAllItems() {
        databaseReference.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                list.clear();
                originalList.clear();
                for (DataSnapshot dataSnapshot : snapshot.getChildren()) {
                    Item item = dataSnapshot.getValue(Item.class);
                    if (item != null) {
                        list.add(item);
                        originalList.add(item);
                    }
                }
                itemAdapter.notifyDataSetChanged();
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                // Handle possible errors.
            }
        });
    }

    private void logoutUser() {
        FirebaseAuth.getInstance().signOut();
        Intent intent = new Intent(this, LoginActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_NEW_TASK);
        startActivity(intent);
        finish();
    }

    private void showToast(String message) {
        Toast.makeText(this, message, Toast.LENGTH_SHORT).show();
    }

    private ArrayList<Item> filter(String query) {
        ArrayList<Item> filteredList = new ArrayList<>();
        for (Item item : originalList) {
            if (item.getName().toLowerCase().contains(query.toLowerCase())) {
                filteredList.add(item);
            }
        }
```

```
        list.clear();
        list.addAll(filteredList);
        itemAdapter.notifyDataSetChanged();
        return filteredList;
    }

    private void goToAcceptedItemFragment() {
        hideViews();
        AcceptedItemFragment acceptedItemFragment = new AcceptedItemFragment();
        getSupportFragmentManager().beginTransaction()
                .replace(R.id.fragment_container, acceptedItemFragment)
                .addToBackStack(null)
                .commit();
    }

    public void hideViews() {
        mainLayout.findViewById(R.id.topAppbar).setVisibility(View.GONE);
        mainLayout.findViewById(R.id.imagegram).setVisibility(View.GONE);
        mainLayout.findViewById(R.id.itemlist).setVisibility(View.GONE);
        mainLayout.findViewById(R.id.search_view).setVisibility(View.GONE);
    }

    private void showViews() {
        mainLayout.findViewById(R.id.topAppbar).setVisibility(View.VISIBLE);
        mainLayout.findViewById(R.id.imagegram).setVisibility(View.VISIBLE);
        mainLayout.findViewById(R.id.itemlist).setVisibility(View.VISIBLE);
        mainLayout.findViewById(R.id.search_view).setVisibility(View.VISIBLE);
    }

    @Override
    public void onBackPressed() {
        FragmentManager fragmentManager = getSupportFragmentManager();
        Fragment currentFragment =
fragmentManager.findFragmentById(R.id.fragment_container);

        // Check if the current fragment is AcceptedItemFragment
        if (currentFragment instanceof VerifyClaimFragment) {
            // Handle back press by showing the main views and popping the fragment
            showViews(); // Ensure main views reappear
            fragmentManager.popBackStack(); // Pop the fragment off the back stack
        } else if (currentFragment instanceof AuthItemHandoverFragment) {
            // Handle back press by showing the main views and popping the fragment
            showViews(); // Ensure main views reappear
            fragmentManager.popBackStack(); // Pop the fragment off the back stack
        } else {
            // Otherwise, call the default back press behavior
            super.onBackPressed();
        }
    }
```

```
    private void listenForNewLostItems() {
        databaseReference.addChildEventListener(new ChildEventListener() {
            @Override
            public void onChildAdded(@NonNull DataSnapshot snapshot, String
previousChildName) {
                // A new lost item has been added
                String itemName = snapshot.child("name").getValue(String.class);
                String location = snapshot.child("location").getValue(String.class);
                String dateTime = snapshot.child("dateTime").getValue(String.class);

                // Send a notification with the lost item details
                String message = "New item reported: " + itemName + " at " + location + " on " +
dateTime;
                NotificationHelper notificationHelper = new NotificationHelper();

notificationHelper.sendNotificationToAuthenticators(AuthenticatorMainActivity.this, "New
Item ", message);
                //notificationHelper.listenForNewItemAdded(getActivity());
//                "Your item has been successfully added."
//                NotificationHelper.sendNotification(AddItemFragment.this, "Lost Item
Submitted", message);
            }

            @Override
            public void onChildChanged(@NonNull DataSnapshot snapshot, String
previousChildName) {
                // Handle item updates, if needed
            }

            @Override
            public void onChildRemoved(@NonNull DataSnapshot snapshot) {}

            @Override
            public void onChildMoved(@NonNull DataSnapshot snapshot, String
previousChildName) {}

            @Override
            public void onCancelled(@NonNull DatabaseError error) {}
        });
    }
}
```

2. AccountActivity.java

```
package com.kundevahal.lostnfound;

public class AccountActivity extends AppCompatActivity {

    private static final String TAG = "AccountActivity";

    private TextView firstname, textfname, middlename, textmname, lastname, textlname,
mobilenumber, textmnname, location, textlocation,profiletxt;
    private Button buttonEditProfile, buttonDeleteAccount;
    private FirebaseAuth auth;
    private FirebaseUser user;
    private ImageButton imageButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_account);

        firstname = findViewById(R.id.firstname);
        textfname = findViewById(R.id.text_firstname);
        middlename = findViewById(R.id.middlename);
        textmname = findViewById(R.id.text_middlename);
        lastname = findViewById(R.id.lastname);
        textlname = findViewById(R.id.text_lastname);
        mobilenumber = findViewById(R.id.mobilenumber);
        textmnname = findViewById(R.id.text_mobilenumber);
        location = findViewById(R.id.location);
        textlocation = findViewById(R.id.text_location);
        buttonEditProfile = findViewById(R.id.button_edit_profile);
        buttonDeleteAccount = findViewById(R.id.button_delete_account);
        imageButton = findViewById(R.id.backarrowbutton);
        profiletxt = findViewById(R.id.profileText);

        auth = FirebaseAuth.getInstance();
        user = auth.getCurrentUser();

        if (user != null) {
            String email = user.getEmail();
            if (email != null) {
                Query userQuery =
FirebaseDatabase.getInstance().getReference().child("Authenticator")
                        .orderByChild("email").equalTo(email);

                userQuery.addValueEventListener(new ValueEventListener() {
                    @Override
                    public void onDataChange(DataSnapshot snapshot) {
                        if (snapshot.exists()) {
                            for (DataSnapshot userSnapshot : snapshot.getChildren()) {
```

```
                    Authenticator authenticator =
userSnapshot.getValue(Authenticator.class);
                    if (authenticator != null) {
                        firstname.setText(authenticator.getFirstname() != null ?
authenticator.getFirstname() : "N/A");
                        middlename.setText(authenticator.getMiddlename() != null ?
authenticator.getMiddlename() : "N/A");
                        lastname.setText(authenticator.getLastname() != null ?
authenticator.getLastname() : "N/A");
                        mobilenumber.setText(authenticator.getContact() != null ?
authenticator.getContact() : "N/A");
                        location.setText(authenticator.getLocation() != null ?
authenticator.getLocation() : "N/A");
                    } else {
                        Log.d(TAG, "Authenticator object is null");
                    }
                }
            } else {
                Log.d(TAG, "Snapshot does not exist");
            }
        }

        @Override
        public void onCancelled(DatabaseError error) {
            Toast.makeText(AccountActivity.this, "Failed to load user info: " +
error.getMessage(), Toast.LENGTH_SHORT).show();
        }
    });

    imageButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            startActivity(new Intent(AccountActivity.this,
AuthenticatorMainActivity.class));
            finish();
        }
    });

    buttonEditProfile.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            firstname.setVisibility(View.GONE);
            middlename.setVisibility(View.GONE);
            lastname.setVisibility(View.GONE);
            mobilenumber.setVisibility(View.GONE);
            location.setVisibility(View.GONE);
            buttonEditProfile.setVisibility(View.GONE);
            buttonDeleteAccount.setVisibility(View.GONE);
            textfname.setVisibility(View.GONE);
            textmname.setVisibility(View.GONE);
```

```
                textlname.setVisibility(View.GONE);
                textmnname.setVisibility(View.GONE);
                textlocation.setVisibility(View.GONE);
                profiletxt.setText("Edit Profile");


                EditAccountFragment editAccountFragment = new EditAccountFragment();
                getSupportFragmentManager().beginTransaction()
                        .replace(R.id.fragment_container, editAccountFragment)
                        .addToBackStack(null)
                        .commit();

            }
        });

        buttonDeleteAccount.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                promptUserForPassword(email);
            }
        });
    } else {
        // Handle case where email is null
        Toast.makeText(this, "User email is null", Toast.LENGTH_SHORT).show();
        startActivity(new Intent(AccountActivity.this, LoginActivity.class));
        finish();
    }
} else {
    // Handle case where user is not authenticated
    Toast.makeText(this, "User not authenticated", Toast.LENGTH_SHORT).show();
    startActivity(new Intent(AccountActivity.this, LoginActivity.class));
    finish();
}

}

private void promptUserForPassword(String email) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Reauthenticate");
    builder.setMessage("Please enter your password to delete your account");

    // Use EditText to allow user to input password
    final EditText input = new EditText(this);
    input.setInputType(InputType.TYPE_CLASS_TEXT |
InputType.TYPE_TEXT_VARIATION_PASSWORD);
    builder.setView(input);

    builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
```

```
            String password = input.getText().toString();
            if (!password.isEmpty()) {
                reauthenticateAndDeleteUserAccount(email, password);
            } else {
                Toast.makeText(AccountActivity.this, "Password cannot be empty",
Toast.LENGTH_SHORT).show();
            }
        }
    });
    builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.cancel();
}
    });

    builder.show();
  }

  private void reauthenticateAndDeleteUserAccount(String email, String password) {
    AuthCredential credential = EmailAuthProvider.getCredential(email, password);
    user.reauthenticate(credential).addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            deleteUserAccount(email);
        } else {
            Toast.makeText(AccountActivity.this, "Re-authentication failed: " +
task.getException().getMessage(), Toast.LENGTH_SHORT).show();
        }
    });
  }

  private void deleteUserAccount(String email) {
    Query userQuery =
FirebaseDatabase.getInstance().getReference().child("Authenticator")
            .orderByChild("email").equalTo(email);

    userQuery.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot snapshot) {
            for (DataSnapshot userSnapshot : snapshot.getChildren()) {
                userSnapshot.getRef().removeValue().addOnCompleteListener(removeTask -> {
                    if (removeTask.isSuccessful()) {
                        // Proceed to delete the user account
                        user.delete().addOnCompleteListener(task -> {
                            if (task.isSuccessful()) {
                                Toast.makeText(AccountActivity.this, "Account deleted",
Toast.LENGTH_SHORT).show();
                                // Redirect to LoginActivity
                                Intent intent = new Intent(AccountActivity.this, LoginActivity.class);
                                startActivity(intent);
```

```
                        finish();
                    } else {
                        Toast.makeText(AccountActivity.this, "Failed to delete account: " +
task.getException().getMessage(), Toast.LENGTH_SHORT).show();
                    }
                });
            } else {
                Toast.makeText(AccountActivity.this, "Failed to delete account data",
Toast.LENGTH_SHORT).show();
            }
        });
    }
}

@Override
public void onCancelled(DatabaseError error) {
    Toast.makeText(AccountActivity.this, "Failed to delete account data",
Toast.LENGTH_SHORT).show();
}
});
}

@Override
public void onBackPressed() {
    super.onBackPressed();
    startActivity(new Intent(AccountActivity.this, AuthenticatorMainActivity.class));
    finish();
}
}
```

3. EditAccountFragment.java

```
package com.kundevahal.lostnfound;

public class EditAccountFragment extends Fragment {

    private EditText firstNameEditText, middleNameEditText, lastNameEditText,
mobileNumberEditText;
    private Button updateButton, cancelButton;
    private FirebaseAuth mAuth;
    private DatabaseReference databaseReference;
    private String email;

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_edit_account, container, false);
```

```
        firstNameEditText = view.findViewById(R.id.editTextFirstName);
        middleNameEditText = view.findViewById(R.id.editTextMiddleName);
        lastNameEditText = view.findViewById(R.id.editTextLastName);
        mobileNumberEditText = view.findViewById(R.id.editTextMobileNumber);
        updateButton = view.findViewById(R.id.buttonUpdate);
        cancelButton = view.findViewById(R.id.buttonCancel);

        mAuth = FirebaseAuth.getInstance();
        email = mAuth.getCurrentUser().getEmail();

        loadUserData();

        updateButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                updateUserData();
            }
        });

        cancelButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(getActivity(), AccountActivity.class));
                getActivity().finish();
            }
        });

        return view;
    }

    private void loadUserData() {
        Query userQuery = FirebaseDatabase.getInstance().getReference("Authenticator")
                .orderByChild("email").equalTo(email);

        userQuery.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                if (snapshot.exists()) {
                    for (DataSnapshot userSnapshot : snapshot.getChildren()) {
                        String firstName = userSnapshot.child("firstname").getValue(String.class);
                        String middleName =
userSnapshot.child("middlename").getValue(String.class);
                        String lastName = userSnapshot.child("lastname").getValue(String.class);
                        String mobileNumber = userSnapshot.child("contact").getValue(String.class);

                        firstNameEditText.setText(firstName);
                        middleNameEditText.setText(middleName);
                        lastNameEditText.setText(lastName);
```

```
                    mobileNumberEditText.setText(mobileNumber);
                }
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {
            Toast.makeText(getActivity(), "Failed to load user data",
Toast.LENGTH_SHORT).show();
        }
    });
  }

  private void updateUserData() {
    String firstName = firstNameEditText.getText().toString().trim();
    String middleName = middleNameEditText.getText().toString().trim();
    String lastName = lastNameEditText.getText().toString().trim();
    String mobileNumber = mobileNumberEditText.getText().toString().trim();

    boolean isValid = true;

    if (TextUtils.isEmpty(firstName) || !firstName.matches("[a-zA-Z ]+")) {
        firstNameEditText.setError("Enter a valid First Name");
        isValid = false;
    } else if (firstName.length()>20) {
        Toast.makeText(getActivity(), "Invalid! More than 20 letters",
Toast.LENGTH_SHORT).show();
    }

    if (TextUtils.isEmpty(middleName) || !middleName.matches("[a-zA-Z ]+")) {
        middleNameEditText.setError("Enter a valid Middle Name");
        isValid = false;
    }
    else if (middleName.length()>20) {
        Toast.makeText(getActivity(), "Invalid! More than 20 letters",
Toast.LENGTH_SHORT).show();
    }

    if (TextUtils.isEmpty(lastName) || !lastName.matches("[a-zA-Z ]+")) {
        lastNameEditText.setError("Enter a valid Last Name");
        isValid = false;
    }
    else if (lastName.length()>20) {
        Toast.makeText(getActivity(), "Invalid! More than 20 letters",
Toast.LENGTH_SHORT).show();
    }

    if (TextUtils.isEmpty(mobileNumber) || mobileNumber.length() != 10 ||
!mobileNumber.matches("[0-9]+")) {
        mobileNumberEditText.setError("Enter a valid Mobile Number (10 digits)");
```

```
            isValid = false;
        }

        if (!isValid) {
            return;
        }

        Query userQuery = FirebaseDatabase.getInstance().getReference("Authenticator")
                .orderByChild("email").equalTo(email);

        userQuery.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                if (snapshot.exists()) {
                    for (DataSnapshot userSnapshot : snapshot.getChildren()) {
                        userSnapshot.getRef().child("firstname").setValue(firstName);
                        userSnapshot.getRef().child("middlename").setValue(middleName);
                        userSnapshot.getRef().child("lastname").setValue(lastName);
                        userSnapshot.getRef().child("contact").setValue(mobileNumber)
                                .addOnCompleteListener(new OnCompleteListener<Void>() {
                                    @Override
                                    public void onComplete(@NonNull Task<Void> task) {
                                        if (task.isSuccessful()) {
                                            Toast.makeText(getActivity(), "Profile updated",
Toast.LENGTH_SHORT).show();
                                            startActivity(new Intent(getActivity(), AccountActivity.class));
                                            getActivity().finish();
                                        } else {
                                            Toast.makeText(getActivity(), "Failed to update profile",
Toast.LENGTH_SHORT).show();
                                        }
                                    }
                                });
                    }
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                Toast.makeText(getActivity(), "Failed to update user data",
Toast.LENGTH_SHORT).show();
            }
        });
    }

}
```

4. VerifyClaimFragment.java

```
package com.kundevahal.lostnfound;

public class VerifyClaimFragment extends Fragment {

    private static final String ARG_ITEM = "item";
    private Item item;
    private TextView itemDateTime;

    public VerifyClaimFragment() {
        // Required empty public constructor
    }

    public static VerifyClaimFragment newInstance(Item item) {
        VerifyClaimFragment fragment = new VerifyClaimFragment();
        Bundle args = new Bundle();
        args.putSerializable(ARG_ITEM, item);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            item = (Item) getArguments().getSerializable(ARG_ITEM);
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_verify_claim, container, false);
        itemDateTime = view.findViewById(R.id.handover_datetime);
        itemDateTime.setOnClickListener(v -> showDateTimePicker());
        return view;
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle
savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

        // Initialize views
        TextView tvItemName = view.findViewById(R.id.tvItemName);
        TextView tvItemLocation = view.findViewById(R.id.tvItemLocation);
        TextView tvItemDateTime = view.findViewById(R.id.tvItemDateTime);
        ImageView imgItemImage = view.findViewById(R.id.imgItemImage);
        ImageView imgAdharImage = view.findViewById(R.id.imgAdharImage);
```

```
        TextView tvDescription = view.findViewById(R.id.tvDescription);
        TextView tvAdhar = view.findViewById(R.id.tvAdhar);
        TextView tvUser = view.findViewById(R.id.tvUser);
        TextView tvPhone = view.findViewById(R.id.tvPhone);
        //TextView tvUserEmail = view.findViewById(R.id.tvUserEmail);
        TextView tvImageType = view.findViewById(R.id.tvImageType);
        Button btnCancel = view.findViewById(R.id.btnCancel);
        Button btnAccept = view.findViewById(R.id.btnAccept);

        // Set item details to views
        tvItemName.setText(item.getName());
        tvItemLocation.setText(item.getLocation());
        tvItemDateTime.setText(item.getDateTime());
        Glide.with(getContext()).load(item.getItemImageUrl()).into(imgItemImage);
        Glide.with(getContext()).load(item.getAdharImageUrl()).into(imgAdharImage);
        tvDescription.setText(item.getDescription());
        tvAdhar.setText(item.getAdharNumber());
        tvUser.setText(item.getUserName());
        tvPhone.setText(item.getPhoneNumber());
        tvImageType.setText(item.getFinalImageType());

        // Handle cancel button click
        btnCancel.setOnClickListener(v -> navigateToMainActivity());

        // Handle accept button click
        btnAccept.setOnClickListener(v -> handleAccept());
    }

    private void handleAccept() {
        // Get the entered date and time
        String datenTime = itemDateTime.getText().toString().trim();

        // Check if the date and time field is empty
        if (datenTime.isEmpty()) {
            Toast.makeText(getContext(), "Please select a date and time before accepting.",
Toast.LENGTH_SHORT).show();
            return; // Exit the method
        }
        new AlertDialog.Builder(getContext())
                .setTitle("Confirm Acceptance")
                .setMessage("Are you sure you want to accept this item?")
                .setPositiveButton("Yes", (dialog, which) -> {

                    DatabaseReference databaseRef =
FirebaseDatabase.getInstance().getReference();
                    DatabaseReference lostItemRef =
databaseRef.child("lost_items").child(item.getItemId());
                    DatabaseReference acceptedItemRef =
databaseRef.child("accepted_items").child(item.getItemId());
```

```
                String dateTime = itemDateTime.getText().toString().trim();
                item.setDateTime(dateTime);

                // Transfer item data to accepted_items
                acceptedItemRef.setValue(item).addOnCompleteListener(task -> {
                    if (task.isSuccessful()) {
                        // Delete the item from lost_items
                        lostItemRef.removeValue().addOnCompleteListener(removeTask -> {
                            if (removeTask.isSuccessful()) {
                                Toast.makeText(getContext(), "Item accepted successfully",
Toast.LENGTH_SHORT).show();

                                Intent intent = new Intent(getContext(),
AuthenticatorMainActivity.class);
                                startActivity(intent);

                                // Finish the current activity to prevent going back to it
                                if (getActivity() != null) {
                                    getActivity().finish();
                                }
                            } else {
                                Toast.makeText(getContext(), "Failed to remove item from
lost_items", Toast.LENGTH_SHORT).show();
                            }
                        });
                    } else {
                        Toast.makeText(getContext(), "Failed to accept item",
Toast.LENGTH_SHORT).show();
                    }
                });
            })
            .setNegativeButton("No", (dialog, which) -> dialog.dismiss())
            .setCancelable(false) // Prevents dismissal on outside touch
            .show();
    }

    private void navigateToMainActivity() {
        Intent intent = new Intent(getContext(), AuthenticatorMainActivity.class);
        startActivity(intent);
    }

    private void showDateTimePicker() {
        final Calendar currentDateTime = Calendar.getInstance(); // Current date and time for
validation
        final Calendar selectedDateTime = Calendar.getInstance();

        // DatePickerDialog to select the date
        DatePickerDialog datePickerDialog = new DatePickerDialog(getActivity(),
                (view, year, month, dayOfMonth) -> {
```

```
                selectedDateTime.set(Calendar.YEAR, year);
                selectedDateTime.set(Calendar.MONTH, month);
                selectedDateTime.set(Calendar.DAY_OF_MONTH, dayOfMonth);

                // TimePickerDialog to select the time
                TimePickerDialog timePickerDialog = new TimePickerDialog(getActivity(),
                        (view1, hourOfDay, minute) -> {
                            selectedDateTime.set(Calendar.HOUR_OF_DAY, hourOfDay);
                            selectedDateTime.set(Calendar.MINUTE, minute);

                            // Validate the selected date and time
                            if (selectedDateTime.before(currentDateTime)) {
                                Toast.makeText(getContext(), "Past date and time are not allowed.",
Toast.LENGTH_SHORT).show();
                                itemDateTime.setText(""); // Reset the field
                            } else {
                                SimpleDateFormat dateFormat = new
SimpleDateFormat("dd/MM/yyyy HH:mm", Locale.getDefault());

itemDateTime.setText(dateFormat.format(selectedDateTime.getTime()));
                            }
                        },
                        selectedDateTime.get(Calendar.HOUR_OF_DAY),
                        selectedDateTime.get(Calendar.MINUTE),
                        false);

                // Allow time selection only if the selected date is today
                if (selectedDateTime.get(Calendar.YEAR) ==
currentDateTime.get(Calendar.YEAR) &&
                        selectedDateTime.get(Calendar.DAY_OF_YEAR) ==
currentDateTime.get(Calendar.DAY_OF_YEAR)) {

timePickerDialog.updateTime(currentDateTime.get(Calendar.HOUR_OF_DAY),
currentDateTime.get(Calendar.MINUTE));
                }

                timePickerDialog.show();
            },
            selectedDateTime.get(Calendar.YEAR),
            selectedDateTime.get(Calendar.MONTH),
            selectedDateTime.get(Calendar.DAY_OF_MONTH));

        // Restrict DatePicker to not allow past dates
        datePickerDialog.getDatePicker().setMinDate(currentDateTime.getTimeInMillis());
        datePickerDialog.show();
    }

}
```

5. AcceptedItemFragment.java

```
package com.kundevahal.lostnfound;

public class AcceptedItemFragment extends Fragment implements
AcceptedItemAdapter.OnItemClickListener {

    private RecyclerView recyclerView;
    private AcceptedItemAdapter adapter;
    private ArrayList<Item> acceptedItems;
    private DatabaseReference databaseReference;
    private ImageButton backButton; // Added ImageButton reference

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_accepted_item, container, false);

        recyclerView = view.findViewById(R.id.recyclerViewAcceptedItems);
        recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));

        acceptedItems = new ArrayList<>();
        adapter = new AcceptedItemAdapter(acceptedItems, this);
        recyclerView.setAdapter(adapter);  // Attach the adapter immediately

        backButton = view.findViewById(R.id.btnBackArrow); // Initialize the back button
        backButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Navigate back to AuthenticatorMainActivity using Intent
                Intent intent = new Intent(getActivity(), AuthenticatorMainActivity.class);
                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_NEW_TASK);
                startActivity(intent);
                getActivity().finish(); // Finish the current activity
            }
        });

        loadAcceptedItems();

        // Handle the back press behavior

requireActivity().getOnBackPressedDispatcher().addCallback(getViewLifecycleOwner(),
new OnBackPressedCallback(true) {
            @Override
            public void handleOnBackPressed() {
                // Navigate back to AuthenticatorMainActivity using Intent
                Intent intent = new Intent(getActivity(), AuthenticatorMainActivity.class);
                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
```

```
Intent.FLAG_ACTIVITY_NEW_TASK);
                startActivity(intent);
                getActivity().finish(); // Finish the current activity
            }
        });

        return view;
    }

    private void loadAcceptedItems() {
        databaseReference = FirebaseDatabase.getInstance().getReference("accepted_items");
        databaseReference.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                acceptedItems.clear();
                for (DataSnapshot dataSnapshot : snapshot.getChildren()) {
                    Item item = dataSnapshot.getValue(Item.class);
                    acceptedItems.add(item);
                }
                adapter.notifyDataSetChanged(); // Notify the adapter about data changes
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                // Handle possible errors.
            }
        });
    }

    @Override
    public void onItemClick(Item item) {
        // Navigate to the handover form fragment with item information
        HandoverFormFragment handoverFormFragment =
HandoverFormFragment.newInstance(item);
        getActivity().getSupportFragmentManager().beginTransaction()
                .replace(R.id.fragment_container, handoverFormFragment)
                .addToBackStack(null)
                .commit();
    }

    @Override
    public void hideRecyclerView() {
        recyclerView.setVisibility(View.GONE); // Hide the RecyclerView
    }
}
```

6. HandoverFormFragment.java

```
package com.kundevahal.lostnfound;

public class HandoverFormFragment extends Fragment {

    private static final String ARG_ITEM = "item";
    private static final int REQUEST_IMAGE_CAPTURE = 1;
    private Item item;
    private EditText userNameEditText, userPhoneEditText;
    private Button confirmHandoverButton, takePhotoButton;
    private ImageView itemImageView, capturedImageView;
    private TextView itemNameTextView, itemLocationTextView, itemDateTimeTextView;
    private ProgressBar progressBar;
    private View formContainer;
    private Bitmap capturedBitmap;

    public static HandoverFormFragment newInstance(Item item) {
        HandoverFormFragment fragment = new HandoverFormFragment();
        Bundle args = new Bundle();
        args.putSerializable(ARG_ITEM, item);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            item = (Item) getArguments().getSerializable(ARG_ITEM);
        }
    }

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_handover_form, container, false);

        // Initialize views
        userNameEditText = view.findViewById(R.id.editTextUserName);
        userPhoneEditText = view.findViewById(R.id.editTextUserPhone);
        confirmHandoverButton = view.findViewById(R.id.buttonConfirmHandover);
        takePhotoButton = view.findViewById(R.id.buttonTakePhoto);
        itemImageView = view.findViewById(R.id.imageViewItem);
        itemNameTextView = view.findViewById(R.id.textViewItemName);
        itemLocationTextView = view.findViewById(R.id.textViewItemLocation);
        itemDateTimeTextView = view.findViewById(R.id.textViewItemDateTime);
        progressBar = view.findViewById(R.id.progressBar);
        formContainer = view.findViewById(R.id.formContainer);
        capturedImageView = view.findViewById(R.id.imageViewCaptured);
```

```
    // Show ProgressBar while data loads
    showLoading(true);

    // Set item details in the UI
    if (item != null) {
        itemNameTextView.setText(item.getName());
        itemLocationTextView.setText(item.getLocation());
        itemDateTimeTextView.setText(item.getDateTime());

        // Load item image using Glide
        Glide.with(this).load(item.getItemImageUrl()).into(itemImageView);

        // Once data is set, hide ProgressBar and show the actual views
        showLoading(false);
    }

    takePhotoButton.setOnClickListener(v -> {
        Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        if (takePictureIntent.resolveActivity(getActivity().getPackageManager()) != null) {
            startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
        }
    });

    confirmHandoverButton.setOnClickListener(v -> {
        showLoading(true); // Show the loading progress bar when confirming the handover
        confirmHandover();
    });

    return view;
}

@Override
public void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK) {
        Bundle extras = data.getExtras();
        capturedBitmap = (Bitmap) extras.get("data");
        capturedImageView.setImageBitmap(capturedBitmap);
        capturedImageView.setVisibility(View.VISIBLE);
    }
}

private void showLoading(boolean isLoading) {
    progressBar.setVisibility(isLoading ? View.VISIBLE : View.GONE);
    formContainer.setVisibility(isLoading ? View.GONE : View.VISIBLE);
}

private void confirmHandover() {
    String userName = userNameEditText.getText().toString().trim();
```

```
        String userPhone = userPhoneEditText.getText().toString().trim();

        // Check if all fields are filled
        if (userName.isEmpty() || userPhone.isEmpty() || capturedBitmap == null) {
            Toast.makeText(getContext(), "Please fill in all fields and take a photo.",
Toast.LENGTH_SHORT).show();
            showLoading(false); // Hide loading if validation fails
            return;
        } else if (userName.length()>20) {
            Toast.makeText(getContext(), "Invaid! Too long person name",
Toast.LENGTH_SHORT).show();
            showLoading(false); // Hide loading if validation fails
            return;
        }

        // Check if the phone number matches
        if (!userPhone.equals(item.getPhoneNumber())) {
            Toast.makeText(getContext(), "Phone number does not match the owner's phone
number.", Toast.LENGTH_SHORT).show();
            showLoading(false); // Hide loading if validation fails
            return; // Exit the method if the phone numbers do not match
        }

        uploadImageAndSaveRecord();
    }

    private void uploadImageAndSaveRecord() {
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        capturedBitmap.compress(Bitmap.CompressFormat.JPEG, 100, baos);
        byte[] data = baos.toByteArray();

        StorageReference storageRef =
FirebaseStorage.getInstance().getReference().child("handover_photos/" +
System.currentTimeMillis() + ".jpg");
        UploadTask uploadTask = storageRef.putBytes(data);

        uploadTask.addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                storageRef.getDownloadUrl().addOnSuccessListener(uri -> {
                    String photoUrl = uri.toString();
                    fetchAcceptedItemDataAndSave(photoUrl);
                });
            } else {
                Toast.makeText(getContext(), "Failed to upload image.",
Toast.LENGTH_SHORT).show();
                showLoading(false); // Hide loading if upload fails
            }
        });
    }
```

```
    private void fetchAcceptedItemDataAndSave(String photoUrl) {
        DatabaseReference acceptedItemRef =
FirebaseDatabase.getInstance().getReference("accepted_items").child(item.getItemId());
        acceptedItemRef.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                if (dataSnapshot.exists()) {
                    // Retrieve data from accepted_items
                    String location = dataSnapshot.child("location").getValue(String.class);
                    String email = dataSnapshot.child("userEmail").getValue(String.class);
                    String phoneNumber =
dataSnapshot.child("phoneNumber").getValue(String.class);
                    String description = dataSnapshot.child("description").getValue(String.class);

                    // Save the handover record with additional data
                    saveRecordToDatabase(photoUrl, location, email, phoneNumber, description);
                } else {
                    Toast.makeText(getContext(), "Failed to fetch accepted item details.",
Toast.LENGTH_SHORT).show();
                    showLoading(false);
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {
                Toast.makeText(getContext(), "Failed to fetch accepted item details.",
Toast.LENGTH_SHORT).show();
                showLoading(false);
            }
        });
    }

    private void saveRecordToDatabase(String photoUrl, String location, String email, String
phoneNumber, String description) {
        DatabaseReference databaseRef =
FirebaseDatabase.getInstance().getReference("handover_records");
        String recordId = databaseRef.push().getKey();

        // Create the handover item to save
        HandoverItem handoverItem = new HandoverItem(recordId, item.getName(),
userNameEditText.getText().toString().trim(), userPhoneEditText.getText().toString().trim(),
photoUrl, location, email, phoneNumber, description);

        if (recordId != null) {
            databaseRef.child(recordId).setValue(handoverItem).addOnCompleteListener(task ->
{
                showLoading(false); // Hide loading after saving
                if (task.isSuccessful()) {
                    Toast.makeText(getContext(), "Handover record saved successfully.",
Toast.LENGTH_SHORT).show();
```

```
                deleteAcceptedItem(); // Call method to delete the accepted item from the list
                Intent intent = new Intent(getContext(), AuthenticatorMainActivity.class);
                startActivity(intent);

                // Finish the current activity to prevent going back to it
                if (getActivity() != null) {
                    getActivity().finish();
                }
            } else {
                Toast.makeText(getContext(), "Failed to save handover record.",
Toast.LENGTH_SHORT).show();
            }
        }).addOnFailureListener(e -> {
            showLoading(false); // Hide loading on failure
            Toast.makeText(getContext(), "Failed to save handover record.",
Toast.LENGTH_SHORT).show();
        });
    }
  }

  private void deleteAcceptedItem() {
    DatabaseReference databaseRef =
FirebaseDatabase.getInstance().getReference("accepted_items").child(item.getItemId());
    databaseRef.removeValue().addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            Toast.makeText(getContext(), "Item removed from accepted list.",
Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(getContext(), "Failed to remove item from accepted list.",
Toast.LENGTH_SHORT).show();
        }
    });
  }
}
```

7. AuthItemHandoverFragment.java

```
package com.kundevahal.lostnfound;

public class AuthItemHandoverFragment extends Fragment {

    private RecyclerView recyclerView;
    private HandoverItemAdapter adapter;
    private ArrayList<HandoverItem> handoverItems;
    private DatabaseReference databaseReference;
    private ImageButton backButton;

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_auth_item_handover, container, false);
        recyclerView = view.findViewById(R.id.recyclerViewHandoverItems);
        recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
        handoverItems = new ArrayList<>();
        adapter = new HandoverItemAdapter(handoverItems, getContext());
        recyclerView.setAdapter(adapter);
        backButton = view.findViewById(R.id.btnBackArrowHandover);
        backButton.setOnClickListener(v -> requireActivity().onBackPressed());
        loadHandoverItems();
        return view;
    }

    private void loadHandoverItems() {
        databaseReference = FirebaseDatabase.getInstance().getReference("handover_records");
        databaseReference.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                handoverItems.clear();
                for (DataSnapshot dataSnapshot : snapshot.getChildren()) {
                    HandoverItem item = dataSnapshot.getValue(HandoverItem.class);
                    if (item != null) {
                        handoverItems.add(item);
                    }
                }
                adapter.notifyDataSetChanged();
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
            }
        });
    }
}
```

Pojo Files :

A POJO (Plain Old Java Object) file is a simple Java class that is used to store and transfer data. It does not contain any business logic or dependencies on specific frameworks.

1. User.java

```
package com.kundevahal.lostnfound;

import java.io.Serializable;

public class Users implements Serializable {

   String firstname, middlename, lastname, email, contact;

   public Users(String firstname, String middlename, String lastname, String contact, String email) {
      this.firstname = firstname;
      this.middlename = middlename;
      this.lastname = lastname;
      this.email = email;
      this.contact = contact;
   }

   public Users() {
   }

   public String getFirstname() {
      return firstname;
   }

   public void setFirstname(String firstname) {
      this.firstname = firstname;
   }

   public String getMiddlename() {
      return middlename;
   }

   public void setMiddlename(String middlename) {
      this.middlename = middlename;
   }

   public String getLastname() {
      return lastname;
   }

   public void setLastname(String lastname) {
      this.lastname = lastname;
```

```
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getContact() {
        return contact;
    }

    public void setContact(String contact) {
        this.contact = contact;
    }
}
```

2. Authenticator.java

```
package com.kundevahal.lostnfound;

public class Authenticator {
    String firstname,middlename,lastname,contact,location,email;


    public Authenticator() {
    }

    public Authenticator(String firstname, String middlename, String lastname, String contact,
String location, String email) {
        this.firstname = firstname;
        this.middlename = middlename;
        this.lastname = lastname;
        this.contact = contact;
        this.location = location;
        this.email = email;
    }

    public String getFirstname() {
        return firstname;
    }

    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }

    public String getMiddlename() {
        return middlename;
```

```
    }

    public void setMiddlename(String middlename) {
        this.middlename = middlename;
    }

    public String getLastname() {
        return lastname;
    }

    public void setLastname(String lastname) {
        this.lastname = lastname;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getLocation() {
        return location;
    }

    public void setLocation(String location) {
        this.location = location;
    }

    public String getContact() {
        return contact;
    }

    public void setContact(String contact) {
        this.contact = contact;
    }
}
```

3. Item.java

```
package com.kundevahal.lostnfound;

import java.io.Serializable;

public class Item implements Serializable {
    private String itemId;
    private String name;
    private String location;
```

```
    private String dateTime;
    private String itemImageUrl;
    private String adharImageUrl;
    private String description;
    private String adharNumber;
    private String userName;
    private String phoneNumber;
    private String userEmail;
    private String finalImageType;

    public Item() {

    }

    public Item(String itemId, String name, String location, String dateTime,
            String itemImageUrl, String adharImageUrl, String itemDescription,
            String adharNumber, String userName, String phoneNumber, String userEmail,
String finalImageType) { // Updated constructor
        this.itemId = itemId;
        this.name = name;
        this.location = location;
        this.dateTime = dateTime;
        this.itemImageUrl = itemImageUrl;
        this.adharImageUrl = adharImageUrl;
        this.description = itemDescription;
        this.adharNumber = adharNumber;
        this.userName = userName;
        this.phoneNumber = phoneNumber;
        this.userEmail = userEmail;
        this.finalImageType = finalImageType;
    }

    // Getters and Setters
    public String getUserEmail() {
        return userEmail;
    }

    public void setUserEmail(String userEmail) {
        this.userEmail = userEmail;
    }

    public String getItemId() {
        return itemId;
    }

    public void setItemId(String itemId) {
        this.itemId = itemId;
    }

    public String getName() {
```

```
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getLocation() {
    return location;
}

public void setLocation(String location) {
    this.location = location;
}

public String getDateTime() {
    return dateTime;
}

public void setDateTime(String dateTime) {
    this.dateTime = dateTime;
}

public String getItemImageUrl() {
    return itemImageUrl;
}

public void setItemImageUrl(String itemImageUrl) {
    this.itemImageUrl = itemImageUrl;
}

public String getAdharImageUrl() {
    return adharImageUrl;
}

public void setAdharImageUrl(String adharImageUrl) {
    this.adharImageUrl = adharImageUrl;
}

public String getDescription() {
    return description;
}

public void setDescription(String description) {
    this.description = description;
}

public String getAdharNumber() {
    return adharNumber;
}
```

```
    public void setAdharNumber(String adharNumber) {
        this.adharNumber = adharNumber;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getPhoneNumber() {
        return phoneNumber;
    }

    public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
    }

    public String getFinalImageType() {
        return finalImageType;
    }

    public void setFinalImageType(String finalImageType) {
        this.finalImageType = finalImageType;
    }
}
```

4. AcceptedItem.java

```
package com.kundevahal.lostnfound;

import java.io.Serializable;

public class AcceptedItem implements Serializable {
    private String adharImageUrl;
    private String adharNumber;
    private String dateTime;
    private String description;
    private String itemId;
    private String itemImageUrl;
    private String location;
    private String name;
    private String userEmail;

    public AcceptedItem() {}
```

```
public String getAdharImageUrl() {
    return adharImageUrl;
}

public void setAdharImageUrl(String adharImageUrl) {
    this.adharImageUrl = adharImageUrl;
}

public String getAdharNumber() {
    return adharNumber;
}

public void setAdharNumber(String adharNumber) {
    this.adharNumber = adharNumber;
}

public String getDateTime() {
    return dateTime;
}

public void setDateTime(String dateTime) {
    this.dateTime = dateTime;
}

public String getDescription() {
    return description;
}

public void setDescription(String description) {
    this.description = description;
}

public String getItemId() {
    return itemId;
}

public void setItemId(String itemId) {
    this.itemId = itemId;
}

public String getItemImageUrl() {
    return itemImageUrl;
}

public void setItemImageUrl(String itemImageUrl) {
    this.itemImageUrl = itemImageUrl;
}

public String getLocation() {
    return location;
```

```
    }

    public void setLocation(String location) {
        this.location = location;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getUserEmail() {
        return userEmail;
    }

    public void setUserEmail(String userEmail) {
        this.userEmail = userEmail;
    }
}
```

5. HandoverItem.java

```
package com.kundevahal.lostnfound;

public class HandoverItem {
    private String id;
    private String itemName;
    private String userName;
    private String userPhone;
    private String photoUrl;
    private String location;
    private String email;
    private String phoneNumber;
    private String description;

    public HandoverItem() {
    }

    public HandoverItem(String id, String itemName, String userName, String userPhone,
String photoUrl, String location, String email, String phoneNumber, String description) {
        this.id = id;
        this.itemName = itemName;
        this.userName = userName;
        this.userPhone = userPhone;
        this.photoUrl = photoUrl;
        this.location = location;
        this.email = email;
```

```
    this.phoneNumber = phoneNumber;
    this.description = description;
}

// Getters and Setters
public String getId() {
    return id;
}

public void setId(String id) {
    this.id = id;
}

public String getItemName() {
    return itemName;
}

public void setItemName(String itemName) {
    this.itemName = itemName;
}

public String getUserName() {
    return userName;
}

public void setUserName(String userName) {
    this.userName = userName;
}

public String getUserPhone() {
    return userPhone;
}

public void setUserPhone(String userPhone) {
    this.userPhone = userPhone;
}

public String getPhotoUrl() {
    return photoUrl;
}

public void setPhotoUrl(String photoUrl) {
    this.photoUrl = photoUrl;
}

public String getLocation() {
    return location;
}

public void setLocation(String location) {
```

```
        this.location = location;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPhoneNumber() {
        return phoneNumber;
    }

    public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }
}
```

Adapter Files :

1. ItemAdapter.java

```
package com.kundevahal.lostnfound;

import java.util.ArrayList;

public class ItemAdapter extends RecyclerView.Adapter<ItemAdapter.ItemViewHolder> {

    private Context context;
    private ArrayList<Item> arrayList;
    private FirebaseAuth auth;

    public ItemAdapter(Context context, ArrayList<Item> arrayList) {
        this.context = context;
        this.arrayList = arrayList;
        this.auth = FirebaseAuth.getInstance();
    }

    @NonNull
    @Override
    public ItemViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        View v = LayoutInflater.from(context).inflate(R.layout.listed_item, parent, false);
        return new ItemViewHolder(v);
    }

    @Override
    public void onBindViewHolder(@NonNull ItemViewHolder holder, int position) {
        Item item = arrayList.get(position);
        holder.itemname.setText(item.getName());
        holder.location.setText(item.getLocation());
        holder.datetime.setText(item.getDateTime());

        // Load image using Glide
        Glide.with(context)
                .load(item.getItemImageUrl())
                .placeholder(R.drawable.baseline_image_24) // Optional placeholder image
                .into(holder.itemImage);

        String currentUserEmail = auth.getCurrentUser() != null ?
auth.getCurrentUser().getEmail() : null;

        // Check if the current user is the owner of the item
        boolean isCurrentUserOwner = currentUserEmail != null &&
currentUserEmail.equals(item.getUserEmail());

        if (isCurrentUserOwner) {
            holder.itemView.setOnClickListener(v -> showDeleteConfirmationDialog(item));
```

```
        } else if (currentUserEmail != null) {
            checkIfAuthenticator(currentUserEmail, holder, item);
        } else {
            holder.itemView.setOnClickListener(null);
        }
    }

    // Method to check if the current user is an authenticator
    private void checkIfAuthenticator(String email, ItemViewHolder holder, Item item) {
        DatabaseReference databaseRef = FirebaseDatabase.getInstance().getReference();
        databaseRef.child("Authenticator").orderByChild("email").equalTo(email)
                .addListenerForSingleValueEvent(new ValueEventListener() {
                    @Override
                    public void onDataChange(@NonNull DataSnapshot snapshot) {
                        if (snapshot.exists()) {
                            holder.itemView.setOnClickListener(v -> {
                                if (context instanceof AuthenticatorMainActivity) {
                                    ((AuthenticatorMainActivity) context).hideViews();
                                }
                                if (context instanceof FragmentActivity) {
                                    FragmentActivity activity = (FragmentActivity) context;
                                    activity.getSupportFragmentManager()
                                            .beginTransaction()
                                            .replace(R.id.fragment_container,
VerifyClaimFragment.newInstance(item))
                                            .addToBackStack(null) // Optional: Add this transaction to the
back stack
                                            .commit();
                                }
                            });
                        } else {
                            holder.itemView.setOnClickListener(null);
                        }
                    }
    }

    private void showDeleteConfirmationDialog(final Item item) {
        new AlertDialog.Builder(context)
                .setTitle("Delete Item")
                .setMessage("Are you sure you want to delete this item?")
                .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        deleteItem(item);
                    }
                })
                .setNegativeButton("No", null)
                .show();
    }
```

```
    private void deleteItem(Item item) {
        DatabaseReference itemRef =
FirebaseDatabase.getInstance().getReference("lost_items").child(item.getItemId());
        itemRef.removeValue();
        arrayList.remove(item);
        notifyDataSetChanged();
    }

    @Override
    public int getItemCount() {
        return arrayList.size();
    }

    public static class ItemViewHolder extends RecyclerView.ViewHolder {
        TextView itemname, location, datetime;
        ImageView itemImage;

        public ItemViewHolder(@NonNull View itemView) {
            super(itemView);
            itemname = itemView.findViewById(R.id.tvitemname);
            location = itemView.findViewById(R.id.tvlocation);
            datetime = itemView.findViewById(R.id.tvdatetime);
            itemImage = itemView.findViewById(R.id.item_image);
        }
    }
}
```

2. UserAcceptedItemsAdapter :

```
package com.kundevahal.lostnfound;

public class UserAcceptedItemsAdapter extends
RecyclerView.Adapter<UserAcceptedItemsAdapter.ViewHolder> {

    private List<AcceptedItem> acceptedItems;
    private Context context;

    public UserAcceptedItemsAdapter(List<AcceptedItem> acceptedItems, Context context) {
        this.acceptedItems = acceptedItems;
        this.context = context;
    }

    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.card_accepted_item, parent, false);
        return new ViewHolder(view);
    }
```

```
    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
        AcceptedItem item = acceptedItems.get(position);
        holder.textViewItemName.setText(item.getName());
        holder.textViewLocation.setText(item.getLocation());
        holder.textViewDateTime.setText(item.getDateTime());

        // Load the item image using Glide
        Glide.with(context).load(item.getItemImageUrl()).into(holder.imageViewItem);
    }

    @Override
    public int getItemCount() {
        return acceptedItems.size();
    }

    public static class ViewHolder extends RecyclerView.ViewHolder {
        public ImageView imageViewItem;
        public TextView textViewItemName;
        public TextView textViewLocation;
        public TextView textViewDateTime;

        public ViewHolder(View itemView) {
            super(itemView);
            imageViewItem = itemView.findViewById(R.id.imageViewItem);
            textViewItemName = itemView.findViewById(R.id.textViewItemName);
            textViewLocation = itemView.findViewById(R.id.textViewLocation);
            textViewDateTime = itemView.findViewById(R.id.textViewDateTime);
        }
    }
}
```

3. HandoverItemAdapter.java

```
package com.kundevahal.lostnfound;

public class HandoverItemAdapter extends
RecyclerView.Adapter<HandoverItemAdapter.HandoverViewHolder> {

    private ArrayList<HandoverItem> handoverItems;
    private Context context;

    public HandoverItemAdapter(ArrayList<HandoverItem> handoverItems, Context context)
{
        this.handoverItems = handoverItems;
        this.context = context;
    }
```

```
@NonNull
    @Override
    public HandoverViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        View view = LayoutInflater.from(context).inflate(R.layout.handover_item_layout,
parent, false);
        return new HandoverViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull HandoverViewHolder holder, int position) {
        HandoverItem item = handoverItems.get(position);
        holder.itemName.setText(item.getItemName());
        holder.userName.setText(item.getUserName());
        holder.userPhone.setText(item.getUserPhone());

        // Load photo using Glide
        Glide.with(context).load(item.getPhotoUrl()).into(holder.itemPhoto);
    }

    @Override
    public int getItemCount() {
        return handoverItems.size();
    }

    public static class HandoverViewHolder extends RecyclerView.ViewHolder {
        ImageView itemPhoto;
        TextView itemName, userName, userPhone;

        public HandoverViewHolder(@NonNull View itemView) {
            super(itemView);
            itemPhoto = itemView.findViewById(R.id.imageViewItemPhoto);
            itemName = itemView.findViewById(R.id.textViewItemName);
            userName = itemView.findViewById(R.id.textViewUserName);
            userPhone = itemView.findViewById(R.id.textViewUserPhone);
        }
    }
}
```

5.3 Testing Approach

Testing is an essential phase in the software development lifecycle to ensure that the Lost n' Found application functions correctly, meets user requirements, and operates efficiently. The testing approach adopted for this project includes unit testing and integration testing to verify both individual components and their interactions. Unit testing focuses on testing individual methods, classes, and functions to ensure correctness, while integration testing ensures that different modules work together seamlessly. A structured testing approach helps identify and resolve errors at an early stage, improving the overall reliability and security of the application.

5.3.1 Unit Testing

Unit testing is the process of validating the correctness of individual functions, classes, and modules in isolation. It ensures that each component behaves as expected under different conditions. In the Lost n' Found app, unit tests are primarily performed on authentication, database operations, form validations, and search functionalities. For instance, login validation, password recovery, and Firebase authentication are tested independently to ensure accuracy.

The key objectives of unit testing include verifying that each method processes input correctly, returns expected outputs, and handles exceptions properly. Additionally, unit tests check for edge cases and boundary conditions, such as empty search inputs, incorrect login credentials, and invalid form submissions.

5.3.2 Integrated Testing

Integration testing is performed after unit testing to validate that different modules, components, and external services work together correctly. Since the Lost n' Found app involves interactions between user authentication, Firebase Realtime Database, image uploads, and the item search feature, integration testing is crucial in ensuring smooth data flow and proper synchronization between these components.

The primary goal of integration testing is to check whether data is correctly transmitted between components, such as verifying that newly submitted lost items are stored in the database and retrieved correctly when searched. Additionally, it ensures that user authentication is properly linked to different functionalities, allowing only authorized users to access specific features.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

The Lost n' Found app makes it easier for people to locate and claim lost items in an organized and efficient way. By using Firebase Realtime Database, Authentication, and Cloud Storage, the app keeps data secure and well-managed, ensuring a smooth experience for users. It simplifies the process of reporting, verifying, and retrieving lost items, reducing the hassle of manual tracking and making the entire system more reliable.

Designed with a user-friendly interface and strong security measures, the app ensures that only authorized users can handle lost and found items. The Authenticator plays a crucial role in verifying claims, making sure that lost belongings are returned to their rightful owners. Additionally, rigorous testing and security protocols contribute to the app's stability and effectiveness in real-world use.

In the end, Lost n' Found provides a trusted and accessible solution for managing lost items. It not only brings convenience to users but also enhances security and efficiency, making it a valuable tool for any community.

6.2 Future Work

Future improvements will enhance the app's functionality and user engagement. One major planned feature is a notification page, where the Authenticator can post updates, announcements, and important information. Users will be able to access these notifications in a dedicated section, ensuring better communication and timely updates.

Additional enhancements include:

- AI-powered image recognition to help match lost and found items more efficiently.
- Chatbot integration to assist users in navigating the app and resolving queries.
- Multi-language support to improve accessibility for a wider audience.

CHAPTER 7

REFERENCES

The References used are as follows :

1. www.tutorialspoint.com
2. www.youtube.com
3. www.w3schools.com
4. www.stackoverflow.com