VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JNANA SANGAMA", BELAGAVI-590018

A MOBILE APPLICATION DEVELOPMENT LABORATORY AND MINI PROJECT (18AIL68)

REPORT ON

LOST FINDER APPLICATION

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Engineering

In

Artificial Intelligence & Machine Learning

By

KARTIK BHATT

1KS20AI015

VIPUL KANT TRIPTHI

1KS20AI044

Under the guidance of

Mr. Abhilash Bhat

Asst. Prof, Dept. Of CSE

Ms. Sahana Sharma M

Asst. Prof, Dept. Of AIML

KSIT

Department of Artificial Intelligence & Machine Learning

K.S. INSTITUTE OF TECHNOLOGY

#14, Raghuvanahalli, Kanakapura Main Road, Bengaluru-560109

K.S. INSTITUTE OF TECHNOLOGY

#14, Raghuvanahalli, Kanakapura Main Road, Bengaluru-560109

Department of Artificial Intelligence & Machine Learning

CERTIFICATE

This is to certify that Mini Project work entitled **"LOST FINDER APPLICATION"** is carried out by **Mr KARTIK BHATT** bearing USN **1KS20AI015** and **Mr VIPUL KANT TRIPATHI** bearing USN **1KS20AI044** bonafidestudent of **K.S. Institute of Technology** in the partial fulfilment for the award of the **Bachelor of Engineering** in **Artificial Intelligence & Machine Learning** ofthe **Visvesvaraya Technological University, Belagavi,** during the year 2022-23.It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The mini project report has been approved as it satisfies the academic requirements inrespect of Mini Project work prescribed for the said degree for the Sixth semester.

Prof. Anu Mathews
Asst. Professor & In-charge HOD,
AIML Department, KSIT, Bangalore.

Dr. Dilip Kumar K
Principal/Director,
KSIT, Bangalore.

Mr. Abhilash Bhat
Asst. Prof, Dept. Of CSE

Ms. Sahana Sharma M
Asst. Prof, Dept. Of AIML

Name of the Examiners

1.

2.

Signature with date

17/7

17/7

ACKNOWLEDGEMENT

I take this opportunity to thank everyone involved in the successful implementation of this mini project. I would like to thank the college for providing me an opportunity to work on the mini project.

I take this opportunity to express my sincere gratitude to my college **K.S. Institute of Technology,** Bengaluru for providing the environment to work on this mini project.

I would like to express my gratitude to our **MANAGEMENT,** K.S. Institute of Technology, Bengaluru, for providing a very good infrastructure and all the support provided for carrying out this mini project work in college.

I would like to express my gratitude to **Dr. K.V.A Balaji, CEO**, K.S. Institute of Technology, Bengaluru, for his valuable guidance.

I would like to express my gratitude to **Dr. Dilip Kumar K, Principal/Director**, K.S. Institute of Technology, Bengaluru, for his continuous support.

I like to extend my gratitude to **Prof. Anu Mathews, Asst. Professor & In-charge HOD** ,Department of Artificial Intelligence & Machine Learning, for providing very good facilities and all the support provided in carrying out this Mini Project successfully.

I also like to thank my Mini Project Coordinators, **Ms. Sahana Sharma M, Asst. Professor,** Department of Artificial Intelligence & Machine Learning and **Mr. Abhilash Bhat, Asst. Professor,** Department of Computer Science and Engineering for their help and support provided to carry out the Mini Project work successfully.

I am also thankful to the teaching and non-teaching staff of Artificial Intelligence & Machine Learning Department, KSIT for the help provided in completing this Mini Project.

KARTIK BHATT

1KS20AI015

VIPUL KANT TRIPATHI

1KS20AI044

ABSTRACT

The Lost Finder Application is an innovative mobile application designed to facilitate the process of finding and returning lost belongings within a college campus. The application serves as a platform for users to post details about items they have lost, which are then made visible to all other users of the app. This enables a collaborative effort among the college community, including students, teaching staff, and non-teaching staff, to help locate and return lost items to their rightful owners.The application operates through two distinct user roles: the admin and the clients. The admin is responsible for overseeing the lost and found department of the college, while clients include students, teaching staff, and non-teaching staff who have lost or found items. The admin ensures the smooth functioning of the app, moderating posts, and facilitating communication between users.

CHAPTER 1

INTRODUCTION

1.1 Overview

The Lost Finder Application is a cutting-edge mobile application designed to revolutionize the process of locating and returning lost items within a college campus. By leveraging the power of technology and community engagement, the application provides a seamless platform for users to report and find lost belongings, fostering a sense of responsibility and collaboration within the college community.

Traditionally, the process of finding lost items on a college campus has been time-consuming and inefficient, often relying on physical notice boards or word-of-mouth communication. The Lost Finder Application aims to streamline this process by offering a centralized and automated system that connects individuals who have lost items with those who have found them.

1.2 Problem Statement

The current manual system for managing lost items on college campuses presents challenges that hinder the timely retrieval and return of belongings. These include a lack of centralized information, limited reach and communication, inefficient record-keeping, time-consuming search processes, a lack of collaboration and responsibility, and ineffective communication channels.

The absence of a centralized platform leads to difficulties in accessing comprehensive and up-to-date information about lost items, resulting in delays and confusion in locating them. Communication methods relying on word-of-mouth or physical posters limit awareness and decrease the chances of finding and returning lost belongings.

Manual record-keeping is prone to errors, redundancy, and loss of information, making it challenging to accurately document lost items. The search process becomes time-consuming and tedious due to the lack of a centralized system, requiring individuals to rely on multiple sources of information.

1.3 Objective

The Lost Finder Application has been developed with several key objectives in mind to address the limitations of the existing manual system for managing lost items on college campuses.

The primary objective of the Lost Finder Application is to streamline the process of reporting lost

items. By providing a user-friendly interface, the application aims to simplify the task of creating detailed posts with accurate descriptions, location information, and contact details. This objective ensures that reporting lost items is an efficient and accessible task for all users, minimizing the effort required to provide essential information about the lost belongings.

Another objective of the Lost Finder Application is to enhance the visibility and reach of lost item reports. In the current manual system, information about lost items is often scattered across physical notice boards or communicated through word-of-mouth. By providing a centralized platform, the application makes the posted data visible to all users of the app. This objective increases the chances of finding lost belongings by engaging a larger audience within the college community, as more people have access to the information and can actively participate in the search and recovery process.

1.4 Android Studio

The Lost Finder Application is developed using Android Studio, a powerful integrated development environment (IDE) specifically designed for Android app development. Android Studio offers a comprehensive set of tools and features that streamline the development process. It includes a user-friendly layout editor for designing the application's interface and a code editor with helpful features like auto-completion and syntax highlighting. Android Studio also provides an Android Emulator for testing the app on virtual devices, ensuring compatibility across different screen sizes and Android versions. Additionally, the IDE offers debugging and profiling tools to identify and resolve issues, optimizing the app's performance. Overall, Android Studio plays a crucial role in the development of the Lost Finder Application, enabling efficient development, testing, and optimization of the Android app.

In summary, Android Studio serves as the primary development environment for the Lost Finder Application. Its comprehensive set of tools, including the layout editor, code editor, Android Emulator, and debugging tools, facilitate the creation of a user-friendly and compatible app. By leveraging the capabilities of Android Studio, the development team can ensure a seamless and efficient development process, resulting in a high-quality and robust mobile application for locating and returning lost items on college campuses.

1.5 Android Studio Architecture

Android architecture contains different number of components to support any android device needs. Android software contains an open-source Linux Kernel having collection of number of C/C++ libraries which are exposed through an application framework services.Among all the components Linux Kernel provides main functionality of operating system functions to smartphones and Dalvik Virtual Machine (DVM) provide platform for running an android application.

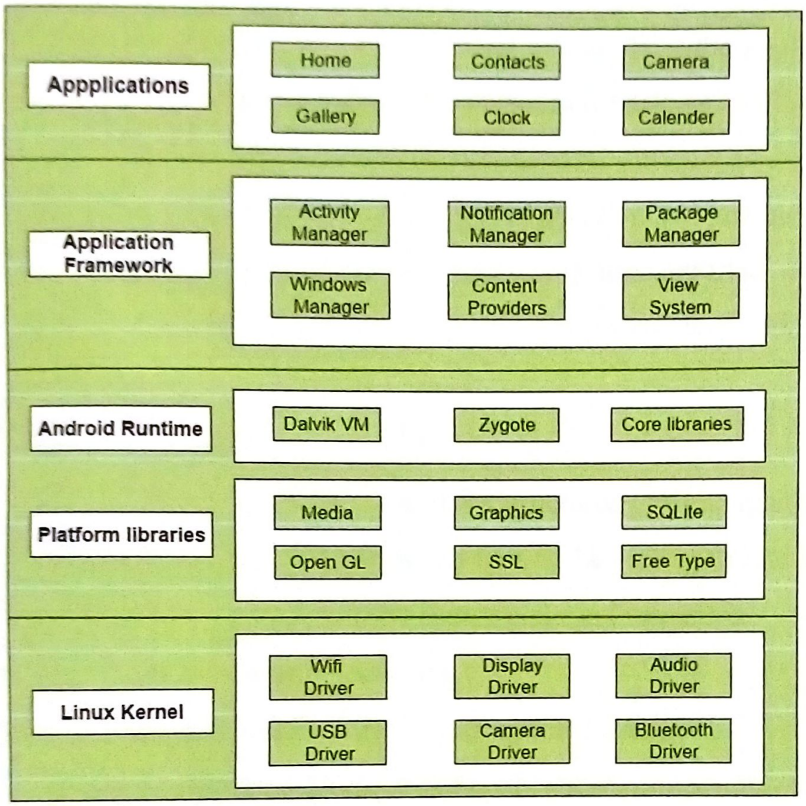

Fig 1.1 Android Architecture

1.6 Methodology

The development of the Lost Finder Application follows a structured methodology to ensure a systematic and successful project execution. The methodology includes the following key stages:

Requirement Analysis: This initial phase involves gathering and understanding the requirements and objectives of the application. It includes identifying the desired features, functionalities, and user expectations to establish a clear project scope.

System Design: The system design stage focuses on planning the architecture and components of the application. It includes creating a blueprint that outlines the user interface (UI), data management, and communication modules, ensuring an organized and efficient system structure.

Development: The development stage involves coding and implementation based on the system

design. It includes writing the necessary code, integrating external libraries or APIs, and creating the user interface as per the design specifications.

Testing: The testing phase is crucial for ensuring the quality and reliability of the application. It involves various testing activities, such as unit testing, integration testing, and user acceptance testing, to identify and fix any bugs or issues.

Deployment and Launch: Once testing is complete, the application is prepared for deployment. This involves generating the final application package (APK) and releasing it for installation on Android devices.

Maintenance and Updates: After the application is launched, maintenance activities ensure its smooth operation. This includes addressing user-reported issues, optimizing performance, and implementing updates or new features based on user feedback and evolving requirements.

1.7 Project Structure

The Lost Finder Application follows a structured project organization to ensure maintainability and modularity. The key directories and files in the project structure include:

app/: Contains the application's source code and resources.

java/: Holds Java source code files.

res/: Stores resources like layouts, strings, and images.

AndroidManifest.xml: Declares the application's components and configurations.

build.gradle: Configures dependencies and build settings for the app module.

Gradle Scripts/: Contains build configuration files for the project and modules.

Other Files: Includes additional files such as ProGuard configuration and Firebase configuration (google-services.json) if applicable.

By adhering to this organized project structure, developers can easily manage and navigate different components, ensuring scalability and maintainability of the Lost Finder Application.

CHAPTER 2

SYSTEM REQUIREMENT

2.1 Introduction

The application aims to provide a convenient and efficient solution for locating lost items on college campuses. It offers a user-friendly platform for users to post information about their lost belongings and facilitates communication between finders and owners for the prompt return of items.

By streamlining the reporting process, enhancing visibility, fostering collaboration, and enabling efficient search and retrieval, the application aims to improve the overall experience of managing lost items within the college community.

With the use of modern technologies and an intuitive interface, the application aims to simplify the process of finding and returning lost belongings, benefiting both users and the college's lost and found department.

2.2 Functional Requirements

Lost Item Posting: Users should be able to create posts providing detailed information about their lost items. This includes descriptions, location details, and contact information to facilitate easy identification and communication.

Lost Item Search and Filtering: The application should allow users to search for lost items based on various criteria such as item category, location, or keywords. Filtering options should be available to refine search results and improve efficiency.

Communication and Messaging: The application should enable direct communication between finders and the original owners of lost items. Users should be able to exchange messages securely within the application, facilitating coordination for the return of belongings.

2.3 Android Studio

Android Studio is an integrated development environment (IDE) specifically designed for Android app development.

It provides a comprehensive set of tools and features to facilitate the development, testing, and debugging of Android applications.

Android Studio includes a user-friendly layout editor for designing the application's user interface (UI) and a code editor with advanced functionalities for efficient coding.

The IDE integrates with Gradle, a build system that automates tasks such as compilation, dependency management, and APK generation.

Android Studio also offers an Android Emulator for testing applications on virtual devices with different configurations, ensuring compatibility and optimal performance.

2.4 Android Studio Emulator

The Android Studio Emulator allows developers to test their applications on virtual Android devices without the need for physical devices.

It simulates various device configurations, screen sizes, resolutions, and Android versions, ensuring compatibility and providing a reliable testing environment.

The emulator enables developers to assess the functionality, performance, and UI responsiveness of the application across different virtual devices.

2.5 Java System Requirement

The Lost Finder Application is developed using Java, a widely used programming language for Android app development.

The system should have the appropriate version of the Java Development Kit (JDK) installed, which provides the necessary tools and libraries for compiling and running Java-based applications.

The JDK version should be compatible with the version of Android Studio being used for development.

The system should meet the minimum hardware and software requirements specified by Oracle for the installed JDK version.

2.6 Objective

Streamline the Reporting Process: The objective is to simplify and streamline the process of reporting lost items, making it user-friendly and efficient.

Enhance Visibility and Reach: The objective is to increase the visibility and reach of lost item reports by providing a centralized platform accessible to all users.

CHAPTER 3

SYSTEM DESIGN

3.1Use Case Model

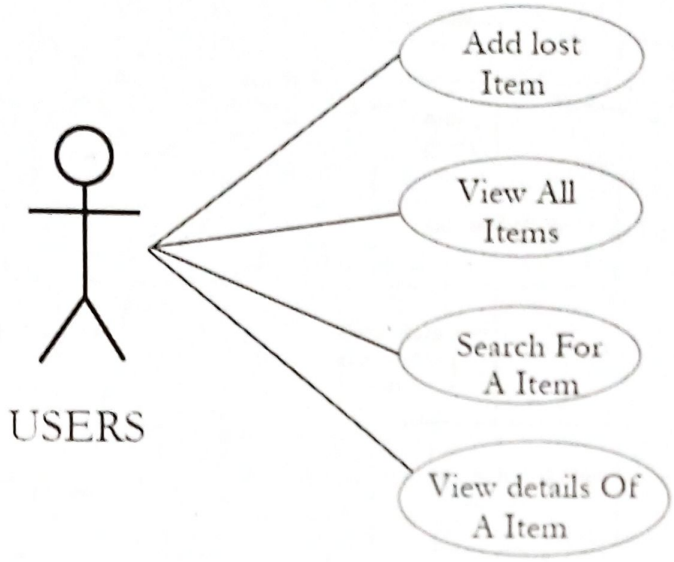

Fig 3.1 Use Case Model For Users

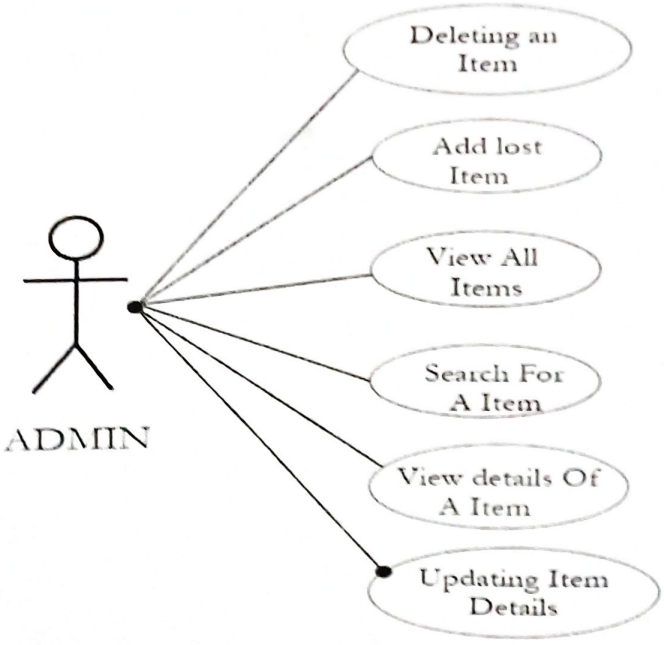

Fig 3.2 Use Case Model For Admins

3.2Data Flow Diagram

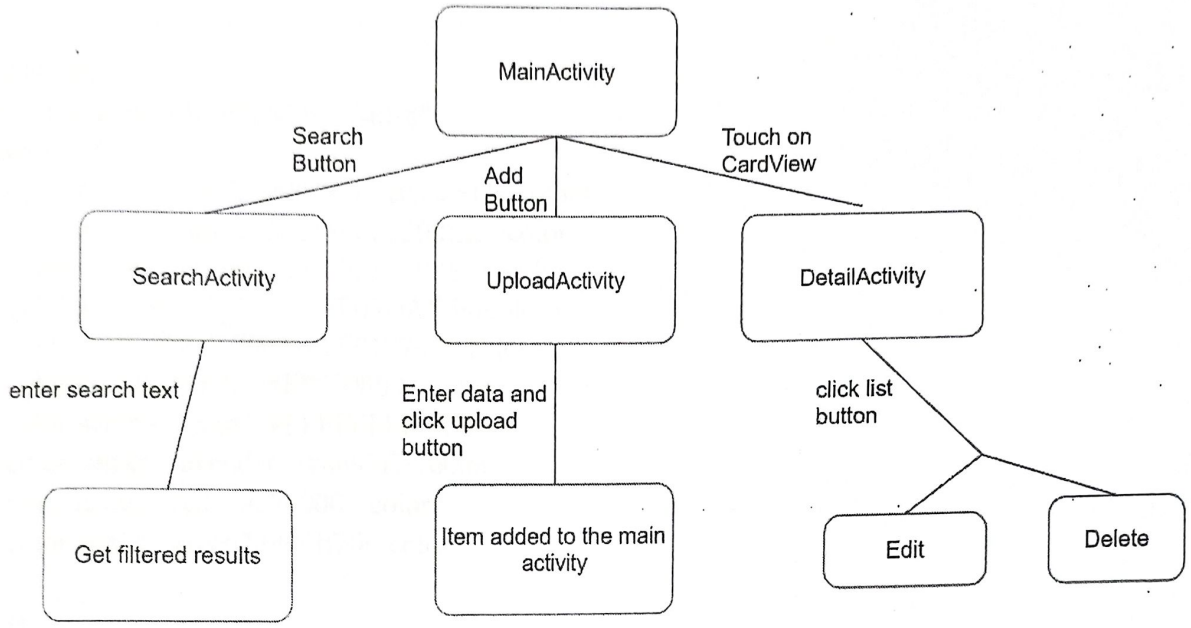

Fig 3.3 Data Flow Diagram Of Application

CHAPTER 4

IMPLEMENTATION

4.1 Overview Of Files In App

Colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
    <color name="lavender">#8692f7</color>
    <color name="red">#c70000</color>
    <color name="green">#00ff00</color>

</resources>
```

Themes.xml

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Theme.Firebase" parent="Theme.MaterialComponents.DayNight.DarkActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/lavender</item>
        <item name="colorPrimaryVariant">@color/lavender</item>
        <item name="colorOnPrimary">@color/white</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_700</item>
        <item name="colorOnSecondary">@color/black</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor">?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>
    <style name="roundedImageViewRounded">
        <item name="cornerFamily">rounded</item>
        <item name="cornerSize">50%</item>
    </style>
</resources>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools">

  <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />

  <application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:supportsRtl="true"
    android:theme="@style/Theme.Firebase"
    tools:targetApi="31">
    <activity
      android:name=".UpdateActivity"
      android:exported="false" />
    <activity
      android:name=".DetailActivity"
      android:exported="false" />
    <activity
      android:name=".UploadActivity"
      android:exported="false" />
    <activity
      android:name=".MainActivity"
      android:exported="true">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>

</manifest>
```

4.2XML Implementation

Activity_main.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"
   tools:context=".MainActivity">

   <androidx.appcompat.widget.SearchView
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_marginTop="16dp"
      android:layout_marginStart="12dp"
      android:layout_marginEnd="12dp"
      android:id="@+id/search"
      app:iconifiedByDefault="false"
      app:searchHintIcon="@null"
      app:queryHint="Search..."
      android:focusable="false"
      app:closeIcon="@drawable/ic_baseline_clear_24"
      app:searchIcon="@drawable/ic_baseline_search_24"
      android:background="@drawable/lavender_border"/>

   <androidx.recyclerview.widget.RecyclerView
      android:layout_marginTop="10dp"
      android:layout_below="@id/search"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:id="@+id/recyclerView"
      android:scrollbars="vertical"/>

   <com.google.android.material.floatingactionbutton.FloatingActionButton
      android:id="@+id/fab"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentBottom="true"
      android:layout_alignParentRight="true"
      android:layout_margin="40dp"
      android:backgroundTint="@color/lavender"
      app:tint = "@color/white"
      android:src="@drawable/ic_baseline_add_24"
      android:contentDescription="TODO" />

</RelativeLayout>
```

activity_upload.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    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:orientation="vertical"
    tools:context=".UploadActivity">

    <androidx.cardview.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="50dp"
        android:layout_marginEnd="20dp"
        android:layout_marginStart="20dp"
        app:cardCornerRadius="30dp"
        app:cardElevation="20dp">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:layout_gravity="center_horizontal"
            android:padding="20dp"
            android:background="@drawable/lavender_border">

            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Upload Data"
                android:textSize="30sp"
                android:textAlignment="center"
                android:textColor="@color/lavender"/>

            <ImageView
                android:layout_width="200dp"
                android:layout_height="200dp"
                android:layout_marginLeft="65dp"
                android:src="@drawable/uploadimg"
                android:id="@+id/uploadImage"
                android:layout_marginTop="10dp"
```

```
    android:scaleType="fitXY"/>

  <EditText
    android:layout_width="match_parent"
    android:layout_height="60dp"
    android:id="@+id/uploadTopic"
    android:background="@drawable/lavender_border"
    android:layout_marginTop="20dp"
    android:padding="16dp"
    android:hint="Name"
    android:gravity="start|center_vertical"
    android:textColor="@color/lavender"/>

  <EditText
    android:layout_width="match_parent"
    android:layout_height="80dp"
    android:id="@+id/uploadDesc"
    android:background="@drawable/lavender_border"
    android:layout_marginTop="20dp"
    android:padding="16dp"
    android:hint="Enter item description"
    android:gravity="start|center_vertical"
    android:textColor="@color/lavender"/>

  <EditText
    android:layout_width="match_parent"
    android:layout_height="60dp"
    android:id="@+id/uploadLang"
    android:background="@drawable/lavender_border"
    android:layout_marginTop="20dp"
    android:padding="16dp"
    android:hint="Contact no."
    android:gravity="start|center_vertical"
    android:textColor="@color/lavender"/>

  <Button
    android:layout_width="match_parent"
    android:layout_height="60dp"
    android:text="Upload Lost"
    android:id="@+id/saveButton"
    android:textSize="18sp"
    android:layout_marginTop="20dp"
    app:cornerRadius = "20dp"/>

</LinearLayout>
```

```
    </androidx.cardview.widget.CardView>

</ScrollView>
```

activity_update.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    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:orientation="vertical"
    tools:context=".UpdateActivity">

    <androidx.cardview.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="50dp"
        android:layout_marginEnd="20dp"
        android:layout_marginStart="20dp"
        app:cardCornerRadius="30dp"
        app:cardElevation="20dp">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:layout_gravity="center_horizontal"
            android:padding="20dp"
            android:background="@drawable/lavender_border">

            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Update Data"
                android:textSize="30sp"
                android:textAlignment="center"
                android:textColor="@color/lavender"/>

            <ImageView
                android:layout_width="match_parent"
                android:layout_height="200dp"
```

```
    android:src="@drawable/uploadimg"
    android:id="@+id/updateImage"
    android:layout_marginTop="10dp"
    android:scaleType="fitXY"/>

<EditText
    android:layout_width="match_parent"
    android:layout_height="60dp"
    android:id="@+id/updateTitle"
    android:background="@drawable/lavender_border"
    android:layout_marginTop="20dp"
    android:padding="16dp"
    android:hint="Name"
    android:gravity="start|center_vertical"
    android:textColor="@color/lavender"/>

<EditText
    android:layout_width="match_parent"
    android:layout_height="80dp"
    android:id="@+id/updateDesc"
    android:background="@drawable/lavender_border"
    android:layout_marginTop="20dp"
    android:padding="16dp"
    android:hint="Item description"
    android:gravity="start|center_vertical"
    android:textColor="@color/lavender"/>

<EditText
    android:layout_width="match_parent"
    android:layout_height="60dp"
    android:id="@+id/updateLang"
    android:background="@drawable/lavender_border"
    android:layout_marginTop="20dp"
    android:padding="16dp"
    android:hint="Contact no."
    android:gravity="start|center_vertical"
    android:textColor="@color/lavender"/>

<Button
    android:layout_width="match_parent"
    android:layout_height="60dp"
    android:text="Update"
    android:id="@+id/updateButton"
    android:textSize="18sp"
    android:layout_marginTop="20dp"
```

```
        app:cornerRadius = "20dp"/>
    </LinearLayout>
  </androidx.cardview.widget.CardView>
</ScrollView>
```

4.3Java Implementation

Upload Item Function: The upload item function allows users to create posts and provide details about their lost items. Users can fill in information such as item description, location, and contact details. This function enables users to report their lost belongings, making them visible to other users of the application.

Delete Item Function: The delete item function allows users to remove their posted lost item from the application. If an item is found or is no longer relevant, users can choose to delete their post, removing it from the list of lost items. This function helps keep the list updated and ensures that only active and relevant lost items are displayed.

Update Item Detail Function: The update item detail function enables users to modify the details of their posted lost items. If there are any changes to the item's description, location, or contact information, users can update the respective fields. This function ensures that the information associated with a lost item remains accurate and up to date.

Filter Search Function: The filter search function allows users to refine their search results based on specific criteria. Users can apply filters such as item category, location, or keywords to narrow down the list of lost items. This function helps users find specific lost items more efficiently by displaying only the relevant results that match their specified criteria.

mainActivity.java

```
package com.example.firebase;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.SearchView;
import androidx.recyclerview.widget.GridLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import java.util.ArrayList;
import java.util.List;
```

```
public class MainActivity extends AppCompatActivity {
    FloatingActionButton fab;
    DatabaseReference databaseReference;
    ValueEventListener eventListener;
    RecyclerView recyclerView;
    List<DataClass> dataList;
    MyAdapter adapter;
    SearchView searchView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        recyclerView = findViewById(R.id.recyclerView);
        fab = findViewById(R.id.fab);
        searchView = findViewById(R.id.search);
        searchView.clearFocus();
        GridLayoutManager gridLayoutManager = new GridLayoutManager(MainActivity.this, 1);
        recyclerView.setLayoutManager(gridLayoutManager);
        AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
        builder.setCancelable(false);
        builder.setView(R.layout.progress_layout);
        AlertDialog dialog = builder.create();
        dialog.show();
        dataList = new ArrayList<>();
        adapter = new MyAdapter(MainActivity.this, dataList);
        recyclerView.setAdapter(adapter);
        databaseReference = FirebaseDatabase.getInstance().getReference("Android Tutorials");
        dialog.show();
        eventListener = databaseReference.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                dataList.clear();
                for (DataSnapshot itemSnapshot: snapshot.getChildren()){
                    DataClass dataClass = itemSnapshot.getValue(DataClass.class);
                    dataClass.setKey(itemSnapshot.getKey());
                    dataList.add(dataClass);
                }
                adapter.notifyDataSetChanged();
                dialog.dismiss();
            }
            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                dialog.dismiss();
            }
        });
```

```
        searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
            @Override
            public boolean onQueryTextSubmit(String query) {
                return false;
            }
            @Override
            public boolean onQueryTextChange(String newText) {
                searchList(newText);
                return true;
            }
        });
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(MainActivity.this, UploadActivity.class);
                startActivity(intent);
            }
        });
    }
    public void searchList(String text){
        ArrayList<DataClass> searchList = new ArrayList<>();
        for (DataClass dataClass: dataList){
            if (dataClass.getDataTitle().toLowerCase().contains(text.toLowerCase())){
                searchList.add(dataClass);
            }
        }
        adapter.searchDataList(searchList);
    }
}
```

UploadActivity.java

```
package com.example.firebase;

import androidx.activity.result.ActivityResult;
import androidx.activity.result.ActivityResultCallback;
import androidx.activity.result.ActivityResultLauncher;
import androidx.activity.result.contract.ActivityResultContracts;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
```

```
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Toast;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import com.google.firebase.storage.UploadTask;
import java.text.DateFormat;
import java.util.Calendar;
public class UploadActivity extends AppCompatActivity {
    ImageView uploadImage;
    Button saveButton;
    EditText uploadTopic, uploadDesc, uploadLang;
    String imageURL;
    Uri uri;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_upload);
        uploadImage = findViewById(R.id.uploadImage);
        uploadDesc = findViewById(R.id.uploadDesc);
        uploadTopic = findViewById(R.id.uploadTopic);
        uploadLang = findViewById(R.id.uploadLang);
        saveButton = findViewById(R.id.saveButton);
        ActivityResultLauncher<Intent> activityResultLauncher = registerForActivityResult(
                new ActivityResultContracts.StartActivityForResult(),
                new ActivityResultCallback<ActivityResult>() {
                    @Override
                    public void onActivityResult(ActivityResult result) {
                        if (result.getResultCode() == Activity.RESULT_OK){
                            Intent data = result.getData();
                            uri = data.getData();
                            uploadImage.setImageURI(uri);
                        } else {
                            Toast.makeText(UploadActivity.this, "No Image Selected",
Toast.LENGTH_SHORT).show();
                        }
                    }
                }
```

```
        );
        uploadImage.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent photoPicker = new Intent(Intent.ACTION_PICK);
                photoPicker.setType("image/*");
                activityResultLauncher.launch(photoPicker);
            }
        });
        saveButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                saveData();
            }
        });
    }
    public void saveData(){
        StorageReference storageReference =
FirebaseStorage.getInstance().getReference().child("Android Images")
                .child(uri.getLastPathSegment());
        AlertDialog.Builder builder = new AlertDialog.Builder(UploadActivity.this);
        builder.setCancelable(false);
        builder.setView(R.layout.progress_layout);
        AlertDialog dialog = builder.create();
        dialog.show();
        storageReference.putFile(uri).addOnSuccessListener(new
OnSuccessListener<UploadTask.TaskSnapshot>() {
            @Override
            public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
                Task<Uri> uriTask = taskSnapshot.getStorage().getDownloadUrl();
                while (!uriTask.isComplete());
                Uri urlImage = uriTask.getResult();
                imageURL = urlImage.toString();
                uploadData();
                dialog.dismiss();
            }
        }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                dialog.dismiss();
            }
        });
    }
    public void uploadData(){
        String title = uploadTopic.getText().toString();
```

```
        String desc = uploadDesc.getText().toString();
        String lang = uploadLang.getText().toString();
        DataClass dataClass = new DataClass(title, desc, lang, imageURL);
        //We are changing the child from title to currentDate,
        // because we will be updating title as well and it may affect child value.
        String currentDate =
    DateFormat.getDateTimeInstance().format(Calendar.getInstance().getTime());
        FirebaseDatabase.getInstance().getReference("Android Tutorials").child(currentDate)
                .setValue(dataClass).addOnCompleteListener(new OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {
                        if (task.isSuccessful()){
                            Toast.makeText(UploadActivity.this, "Saved", Toast.LENGTH_SHORT).show();
                            finish();
                        }
                    }
                }).addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Toast.makeText(UploadActivity.this, e.getMessage().toString(),
    Toast.LENGTH_SHORT).show();
                    }
                });
        }
    }
```

UpdateActivity.java

```
package com.example.firebase;
import androidx.activity.result.ActivityResult;
import androidx.activity.result.ActivityResultCallback;
import androidx.activity.result.ActivityResultLauncher;
import androidx.activity.result.contract.ActivityResultContracts;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Toast;
```

```
import com.bumptech.glide.Glide;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import com.google.firebase.storage.UploadTask;
public class UpdateActivity extends AppCompatActivity {
    ImageView updateImage;
    Button updateButton;
    EditText updateDesc, updateTitle, updateLang;
    String title, desc, lang;
    String imageUrl;
    String key, oldImageURL;
    Uri uri;
    DatabaseReference databaseReference;
    StorageReference storageReference;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_update);
        updateButton = findViewById(R.id.updateButton);
        updateDesc = findViewById(R.id.updateDesc);
        updateImage = findViewById(R.id.updateImage);
        updateLang = findViewById(R.id.updateLang);
        updateTitle = findViewById(R.id.updateTitle);
        ActivityResultLauncher<Intent> activityResultLauncher = registerForActivityResult(
                new ActivityResultContracts.StartActivityForResult(),
                new ActivityResultCallback<ActivityResult>() {
                    @Override
                    public void onActivityResult(ActivityResult result) {
                        if (result.getResultCode() == Activity.RESULT_OK){
                            Intent data = result.getData();
                            uri = data.getData();
                            updateImage.setImageURI(uri);
                        } else {
                            Toast.makeText(UpdateActivity.this, "No Image Selected",
Toast.LENGTH_SHORT).show();
                        }
                    }
                }
        );
```

```
        Bundle bundle = getIntent().getExtras();
        if (bundle != null){
            Glide.with(UpdateActivity.this).load(bundle.getString("Image")).into(updateImage);
            updateTitle.setText(bundle.getString("Title"));
            updateDesc.setText(bundle.getString("Description"));
            updateLang.setText(bundle.getString("Language"));
            key = bundle.getString("Key");
            oldImageURL = bundle.getString("Image");
        }
        databaseReference = FirebaseDatabase.getInstance().getReference("Android
Tutorials").child(key);
        updateImage.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent photoPicker = new Intent(Intent.ACTION_PICK);
                photoPicker.setType("image/*");
                activityResultLauncher.launch(photoPicker);
            }
        });
        updateButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                saveData();
                Intent intent = new Intent(UpdateActivity.this, MainActivity.class);
                startActivity(intent);
            }
        });
    }
    public void saveData(){
        storageReference = FirebaseStorage.getInstance().getReference().child("Android
Images").child(uri.getLastPathSegment());
        AlertDialog.Builder builder = new AlertDialog.Builder(UpdateActivity.this);
        builder.setCancelable(false);
        builder.setView(R.layout.progress_layout);
        AlertDialog dialog = builder.create();
        dialog.show();
        storageReference.putFile(uri).addOnSuccessListener(new
OnSuccessListener<UploadTask.TaskSnapshot>() {
            @Override
            public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
                Task<Uri> uriTask = taskSnapshot.getStorage().getDownloadUrl();
                while (!uriTask.isComplete());
                Uri urlImage = uriTask.getResult();
                imageUrl = urlImage.toString();
                updateData();
```

```
            dialog.dismiss();
        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            dialog.dismiss();
        }
    });
}
public void updateData(){
    title = updateTitle.getText().toString().trim();
    desc = updateDesc.getText().toString().trim();
    lang = updateLang.getText().toString();
    DataClass dataClass = new DataClass(title, desc, lang, imageUrl);
    databaseReference.setValue(dataClass).addOnCompleteListener(new
OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void> task) {
            if (task.isSuccessful()){
                StorageReference reference =
FirebaseStorage.getInstance().getReferenceFromUrl(oldImageURL);
                reference.delete();
                Toast.makeText(UpdateActivity.this, "Updated", Toast.LENGTH_SHORT).show();
                finish();
            }
        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            Toast.makeText(UpdateActivity.this, e.getMessage().toString(),
Toast.LENGTH_SHORT).show();
        }
    });
}
}
```

CHAPTER 5

SCREENSHOTS

5.1 Home Page

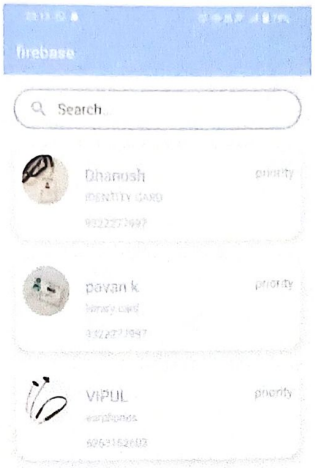

Fig 5.1 MainActivity

5.2 Content Page

Fig 5.2 a.Upload Lost Item

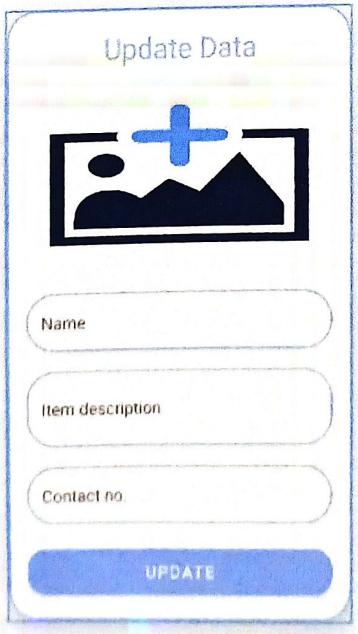

Fig 5.3 b. Update Item Detail

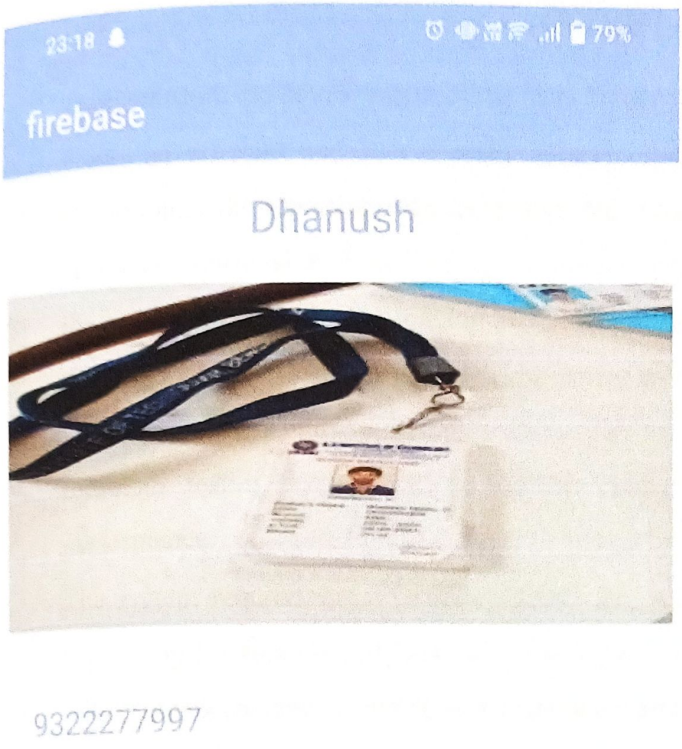

Fig 5.3 Item Detail

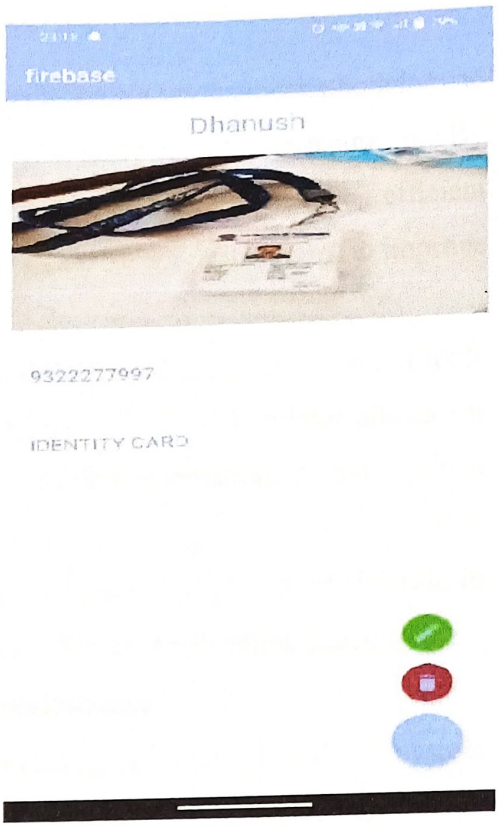

Fig 5.4 Options To Delete And Edit Item

CONCLISION

In conclusion, the development of the Lost Finder Application follows a systematic methodology to create an efficient and user-friendly solution for locating lost items on college campuses. By streamlining the reporting process, enhancing visibility, fostering collaboration, enabling efficient search and retrieval, and facilitating communication between users, the application aims to improve the overall experience of managing lost belongings.

The use of Android Studio as the development environment provides a comprehensive set of tools and features for building the application. The integration with the Android Emulator allows for thorough testing across various device configurations, ensuring compatibility and optimal performance.

The system requirements, including hardware, software, and connectivity, play a crucial role in supporting the functionality and seamless operation of the Lost Finder Application. Meeting these requirements ensures a smooth user experience and reliable performance.

The methodology employed in the development process, including requirement analysis, system design, development, testing, deployment, and maintenance, ensures a structured and systematic approach. This allows for efficient project execution, bug resolution, and adaptation to evolving user needs.

In conclusion, the Lost Finder Application, developed with a clear methodology and in adherence to system requirements, aims to provide a convenient and effective platform for users to locate and recover their lost items on college campuses. By leveraging modern technologies and emphasizing user satisfaction, the application enhances the lost and found process, benefiting both users and the college community as a whole.

FUTURE SCOPE AND ENHANCEMENT

The Lost Finder Application holds immense potential for future enhancements and expansion to further enhance its functionality and user experience. Some potential areas for future development and improvement include:

Integration of Google Maps: One significant enhancement would be the integration of Google Maps to enable real-time tracking of lost items. This feature would allow users to view the live location of their lost belongings on a map, increasing the chances of locating and retrieving the items promptly. Integrating Google Maps would provide users with a visual representation of the item's location, enhancing the overall search and recovery process.

Direct Communication within the App: Another valuable enhancement would be to enable direct communication between finders and the original owners of lost items through the Lost Finder Application itself. By incorporating secure messaging features within the app, users would be able to connect and communicate directly, facilitating smoother coordination and ensuring privacy. This would eliminate the need for users to rely on external communication channels, providing a seamless and convenient communication platform.

Improved Search Filters and Recommendations: Enhancing the search functionality by implementing advanced filters and recommendations would enhance the accuracy and relevance of search results. Users could refine their search based on specific criteria such as item category, date, or proximity, making it easier to find lost belongings quickly. Additionally, incorporating recommendation algorithms could suggest similar lost items or relevant matches, further enhancing the search experience.

Enhanced User Interface and Design: Continual improvement of the user interface (UI) and design elements would contribute to a more visually appealing and user-friendly application. Regular updates to the UI based on user feedback and emerging design trends would enhance usability and ensure a seamless user experience.

Integration with Lost Item Tracking Technologies: Exploring integration with emerging technologies such as RFID (Radio Frequency Identification) or GPS (Global Positioning System) tags would allow for more accurate tracking and identification of lost items. By incorporating these technologies, users could easily tag their belongings, enabling efficient tracking and monitoring within the Lost Finder Application.

Integration with College Systems: Integration with existing college systems, such as student

databases or campus security systems, would enhance the accuracy and reliability of the Lost Finder Application. This integration could facilitate automatic notifications to college authorities, improving coordination and expediting the process of finding and returning lost items.

These potential enhancements would further strengthen the Lost Finder Application's functionality, user experience, and overall effectiveness in locating and returning lost belongings. By continually incorporating new features and leveraging emerging technologies, the application would remain at the forefront of providing a reliable and efficient solution for managing lost items on college campuses.

REFERENCES

1. Android Studio Documentation. Retrieved from: https://developer.android.com/studio/documentation
2. Firebase Documentation. Retrieved from: https://firebase.google.com/docs
3. Gradle Build System Documentation. Retrieved from: https://docs.gradle.org/
4. Android Developer Documentation. Retrieved from: https://developer.android.com/docs
5. Firebase Realtime Database: Android Documentation. Retrieved from: https://firebase.google.com/docs/database/android/start