CHAPTER 1

INTRODUCTION

1.1 OBJECTIVE

The objective of this project is to develop the "ReMinded!" Android application, which simplifies location management and provides timely reminders. By integrating text-to-speech, speech-to-text, data storage, and location tracking, the app aims to enhance user experience and convenience. The objective is to enable users to easily locate items by providing audio feedback, store and retrieve information about item locations, reminders, and other details securely, and deliver location-based reminders for essential tasks or actions. The project focuses on developing a comprehensive Android app that promotes accessibility, organization, and productivity by leveraging advanced technologies in a user-friendly manner. It provides accessibility features as well.

1.2 INTRODUCTION TO ANDROID STUDIO

Android Studio is an authority incorporated advancement condition (IDE) for an android application improvement, in the light of the android Studio is planned explicitly for the android improvement accessibility for the download on Windows & Linux, android application being Google's essential development environment for the local android app advancement. Android Studio also offers adaptable Grade-based form framework, the code formats to enable the user to fabricate the regular android app highlights & the rich format editorial manager with the help for the intuitive topic altering & also worked help for the Google Cloud Platform, making it simple to coordinate with the Google Cloud Messaging android app Engine & considerably more android Studio as good as ever interface structure point of view where you can see the interface you are taking a shot at and its related segments. android Studio give away different UI instruments to help you with making designs, executing style subjects, & making realistic or content assets for your android application. The android manufacture framework the toolbox you use to assemble, test, run & bundle your android applications. The construct framework can keep running as a coordinated android application from the android studio menu & freely from the direction line.

1.3 INTRODUCTION TO DATA STORAGE

Data storage plays a crucial role in the "ReMinded!" Android application. It enables the secure and efficient storage and retrieval of information related to item locations, reminders, and other relevant details. By implementing data storage mechanisms, such as databases, the app ensures that users can store and access their data reliably. The data storage component facilitates seamless organization, retrieval, and management of information, empowering users to efficiently track their belongings and receive timely reminders. It forms the foundation for the app's functionality, allowing users to store and retrieve data conveniently, enhancing the overall user experience and productivity.

1.4. PRODUCT OVERVIEW

The "ReMinded!" Android application is a comprehensive solution for location management and reminders. The project integrates advanced technologies to enhance user experience and convenience. The app incorporates functionalities such as text-to-speech conversion, speech-to-text conversion, data storage, and location tracking.The objective of the application is to simplify the process of locating items by providing audio feedback on their locations through text-to-speech conversion. Users can store information about item locations and retrieve them by issuing voice commands, enhancing accessibility and hands-free operation.The app utilizes a database to securely store and retrieve data, ensuring that information about item locations, reminders, and other details is readily accessible. The database implementation leverages the Room Persistence Library, which simplifies database operations and ensures data integrity.Location-based reminders are a key feature of the app. By leveraging GPS or map detection, the application prompts users with reminders for essential tasks or actions associated with specific locations. This feature prevents forgetfulness and ensures timely completion of important tasks.The development of the "ReMinded!" app takes place in Android Studio, the official IDE for Android application development. Android Studio provides a rich development environment with features such as a visual layout editor, powerful code editor, debugging tools, and seamless integration with the Android SDK.Overall, the "ReMinded!" Android application aims to provide users with a convenient and intuitive tool for location management, information organization, and timely reminders. By combining advanced technologies, the project strives to enhance user convenience, productivity, and accessibility.

CHAPTER 2

REQUIREMENT SPECIFICATION

2.1 PROJECT REQUIREMENTS

The package is designed such that users with an Android phone having minimum configuration can also use it. It does not require complex computing.

The App requires a simple daily use android phone in which this app can be installed and then user can run it. For a developer it is required to install android studio or IntelliJ Application software which is used to design the app using Xml and Java. Data storage is used as backend to store the data.

.

2.2 HARDWARE REQUIREMENTS

- Memory (Primary): 256 GB M.2 PCle NVMe SSD
- Hard Disk: 1TB 7200 rpm
- Processor: 8th Generation Intel Core i5
- RAM: 8 GB
- Graphics: RADEON / NVIDIA
- Monitor resolution - A colour monitor with a minimum resolution

2.3 SOFTWARE REQUIREMENTS

- Operating System: Windows 10
- User Interface: XML
- Programming Language: Java
- Application: Android Studio / IntelliJ
- Local Data Storage

CHAPTER 3

SYSTEM DESIGN

3.1 BASIC LAYOUT

- Home Page for users
- Add Items Page
- View Items Page
- Status Page
- Notification bar for reminders

3.2 ARCHITECTURE

The given project is an Android application called "Reminded." It allows users to keep track of items and their corresponding locations. The app provides features for adding, searching, and managing items, as well as a voice recognition capability for searching items using speech input.When the application starts, the onCreate method is called, where the layout is initialized and the necessary views and variables are set up. The SharedPreferences is used to store and retrieve the item data.The displayItems method populates the UI with the stored items. Each item is represented by a custom layout containing the item name, location, and buttons for text-to-speech and deletion. The searchItem method filters the items based on the user's search query and updates the UI accordingly.Users can delete individual items by confirming a delete action through an alert dialog, or they can delete all items at once. The saveItems method is responsible for storing the updated item data in the SharedPreferences.The app also features text-to-speech functionality using the Android TextToSpeech class. When the user clicks the speaker button for an item, the corresponding item's name and location are spoken out loud.Additionally, the app includes voice search functionality. Clicking the microphone button triggers the speech recognition intent, which listens for user input. The recognized speech is then used to populate the search bar.In summary, the "Reminded" app provides a convenient way for users to manage and search their items and locations. It utilizes shared preferences for data storage, implements various UI components for item display and interaction, and incorporates text-to-speech and speech recognition capabilities to enhance the user experience.

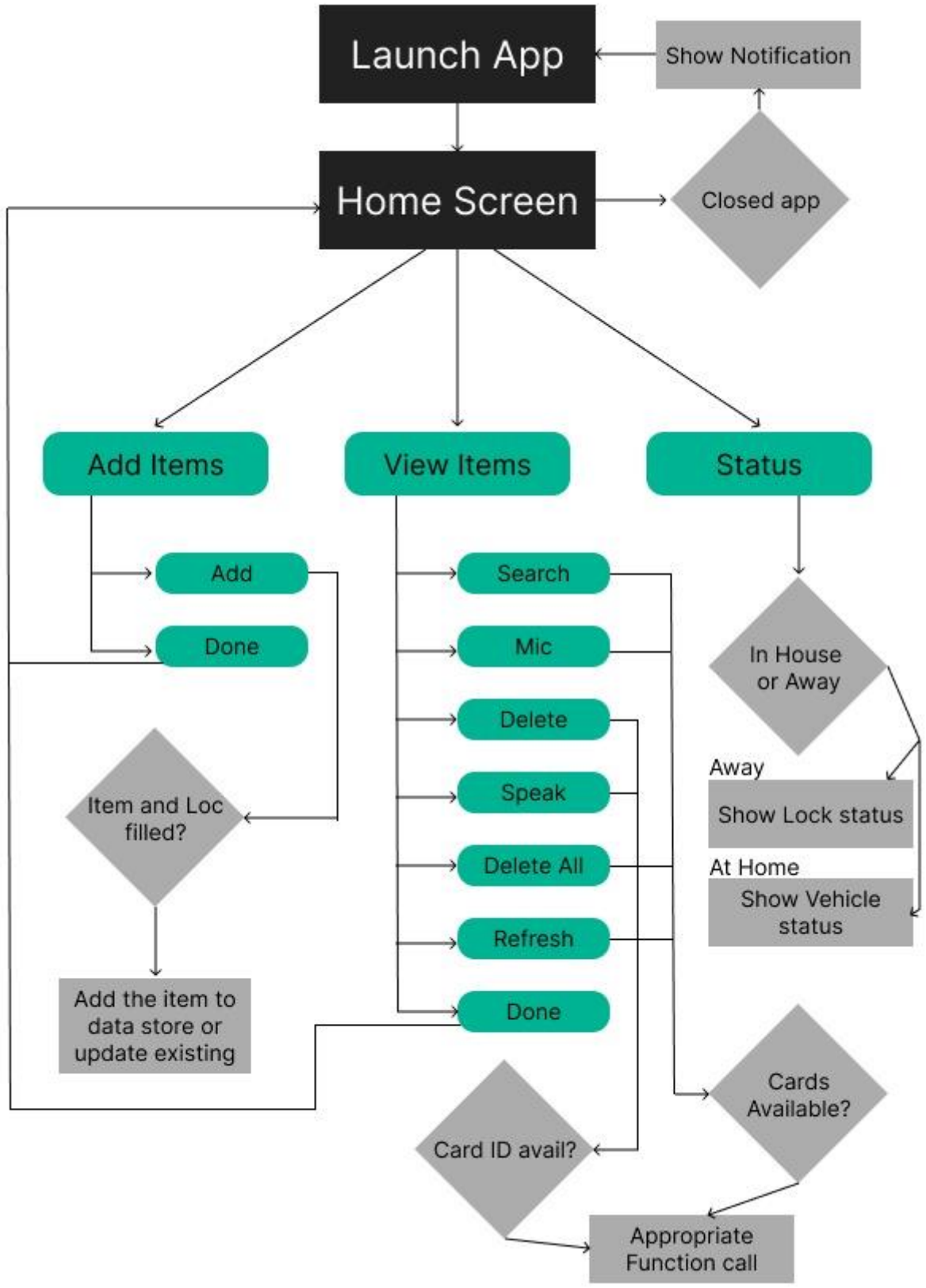

Figure 3.2.1 Flowchart

CHAPTER 4

IMPLEMENTATION

4.1 ALGORITHM – INTRODUCTION

The Reminded algorithm is an intelligent and user-friendly solution integrated into the Reminded Android application, aiming to simplify the management and retrieval of items and their corresponding locations. By incorporating a range of features, this algorithm provides a seamless and intuitive experience for users.At its core, the algorithm utilizes the SharedPreferences mechanism available in the Android platform to store and retrieve item data persistently. This ensures that users can maintain their item list across multiple sessions, guaranteeing data integrity and consistency. By leveraging SharedPreferences, the algorithm enables the app to store and retrieve item names and locations efficiently.One of the key functionalities of the Reminded algorithm is the dynamic population of the user interface. The algorithm retrieves the stored item data and dynamically generates custom layouts to display each item. This approach allows for a visually appealing and organized representation of the user's items, enhancing the overall user experience.To facilitate quick and efficient item retrieval, the algorithm incorporates a search functionality. Users can enter search queries in the provided EditText, and the algorithm filters the items based on the entered text. This feature enables users to effortlessly locate specific items or groups of items, even in large item lists.To enhance accessibility and convenience, the algorithm integrates text-to-speech capabilities using the Android TextToSpeech class. Users can simply tap the speaker button associated with each item, and the algorithm converts the item's details into speech, which is then played back to the user. This feature is particularly useful for individuals with visual impairments or those who prefer auditory information.Additionally, the Reminded algorithm embraces speech recognition capabilities through the Android RecognizerIntent. By invoking the voice search functionality, users can perform item searches by speaking their queries instead of manually typing them. The algorithm captures the spoken words, converts them into text, and populates the search bar accordingly, simplifying the search process and providing a hands-free experience.The algorithm also incorporates user-friendly features such as confirmation dialogs. When users attempt to delete an item or clear the entire item list, a confirmation dialog is presented, ensuring that users can proceed with confidence or cancel the action if needed. This prevents accidental deletions and provides a safety net for users when managing their items.In summary, the Reminded algorithm serves as the backbone of the Reminded Android application, offering a robust and intelligent solution for item management. With its seamless data storage, dynamic UI population, search functionality, text-to-speech integration, speech recognition capabilities, and user-friendly features, the algorithm enhances the overall usability, accessibility, and convenience of the Reminded app, empowering users to efficiently manage and retrieve their items with ease.

4.2 PSEUDO CODE

4.2.1 MAIN ACTIVITY

```
// Making Notification Channels
public class MainActivity extends Activity {

    private static final int NOTIFICATION_ID = 1;
    private static final String CHANNEL_ID = "RemindedChannel";

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

        Button buttonAddItem = findViewById(R.id.buttonAddItem);
        Button buttonViewItems = findViewById(R.id.buttonViewItems);

        buttonAddItem.setOnClickListener(v -> {
            // Start the "AddItems" activity
            Intent = new Intent(MainActivity.this, AddItemsActivity.class);
            startActivity(intent);
        });

        buttonViewItems.setOnClickListener(v -> {
            // Start the "ViewItems" activity
            Intent = new Intent(MainActivity.this, ViewItemsActivity.class);
            startActivity(intent);
        });
    }
    public void openStatusActivity(View view) {
        Intent = new Intent(MainActivity.this, statusloc.class);
        startActivity(intent);


}

 private void hideNotification() {


NotificationManager =
                (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
        notificationManager.cancel(NOTIFICATION_ID);
    }

    private void createNotificationChannel() {
        CharSequence name = "Reminded Channel";
```

```
    String description = "Channel for Reminded notifications";
        int importance = NotificationManager.IMPORTANCE_DEFAULT;
        NotificationChannel channel = new NotificationChannel(CHANNEL_ID, name,
        importance);
        channel.setDescription(description);

        NotificationManager =
            getSystemService(NotificationManager.class);
        notificationManager.createNotificationChannel(channel);
      }
  }
```

4.2.2 ADD ITEMS ACTIVITY

```
  //AddItemsActivity.java

   private void addItem() {
        String itemName = editTextItemName.getText().toString().trim();
        String location = editTextLocation.getText().toString().trim();

        if (itemName.isEmpty()) {
           editTextItemName.setError("Please enter an item name");
           editTextItemName.requestFocus();
           return;
        }

        if (location.isEmpty()) {
           editTextLocation.setError("Please enter a location");
           editTextLocation.requestFocus();
           return;
        }

        SharedPreferences = getSharedPreferences("Items", MODE_PRIVATE);
        SharedPreferences.Editor editor = sharedPreferences.edit();

        int itemCount = sharedPreferences.getInt("itemCount", 0);

        editor.putString("itemName_" + itemCount, itemName);
        editor.putString("location_" + itemCount, location);
        editor.putInt("itemCount", itemCount + 1);

        editor.apply();

        Toast.makeText(this, "Item added successfully", Toast.LENGTH_SHORT).show();

        editTextItemName.getText().clear();
        editTextLocation.getText().clear();
      }
  }
```

4.2.3 VIEW ITEMS ACTIVITY

```
public class ViewItemsActivity extends Activity {

    private static final int REQUEST_CODE_SPEECH_INPUT = 100;
    private LinearLayout linearItemContainer;
    private SharedPreferences;
    private EditText editTextSearch;

    private ArrayList<String> itemNames;
    private ArrayList<String> locations;
    private TextToSpeech;
    private boolean isSpeaking;
        displayItems();

        editTextSearch.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int after) {
                // Not used
            }

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {
                // Not used
            }

            @Override
            public void afterTextChanged(Editable s) {
                searchItem();
            }
        });

        buttonDeleteAll.setOnClickListener(v -> showDeleteAllConfirmation());

        buttonRefresh.setOnClickListener(v -> refreshItems());

        buttonDone.setOnClickListener(v -> finish());
    }

    private void displayItems() {
        linearItemContainer.removeAllViews();

        for (int i = 0; i < itemNames.size(); i++) {
            final int position = i;

            String itemName = itemNames.get(i);
            String location = locations.get(i);
```

```
LinearLayout.LayoutParams layoutParams = new LinearLayout.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
    ViewGroup.LayoutParams.WRAP_CONTENT);
      layoutParams.setMargins(0, 0, 0, 16);

      LinearLayout itemLayout = new LinearLayout(this);
      itemLayout.setLayoutParams(layoutParams);
      itemLayout.setOrientation(LinearLayout.VERTICAL);
      itemLayout.setBackgroundResource(R.drawable.item_background);

      LinearLayout.LayoutParams textViewParams = new LinearLayout.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
    ViewGroup.LayoutParams.WRAP_CONTENT);

      TextView textViewItemName = new TextView(this);
      textViewItemName.setLayoutParams(textViewParams);
      String displayText="ITEM: "+itemName;
      textViewItemName.setText(displayText);
      textViewItemName.setTextSize(20);
      textViewItemName.setTypeface(Typeface.DEFAULT_BOLD);
      textViewItemName.setPadding(16, 16, 16, 0);

      TextView textViewLocation = new TextView(this);
      displayText="LOC: "+location;
      textViewLocation.setLayoutParams(textViewParams);
      textViewLocation.setText(displayText);
      textViewLocation.setTextSize(16);
      textViewLocation.setPadding(16, 0, 16, 0);

      LinearLayout buttonLayout = new LinearLayout(this);
      LinearLayout.LayoutParams buttonLayoutParams = new LinearLayout.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
    ViewGroup.LayoutParams.WRAP_CONTENT);
      buttonLayoutParams.setMargins(0, 16, 0, 16);
      buttonLayout.setLayoutParams(buttonLayoutParams);
      buttonLayout.setOrientation(LinearLayout.HORIZONTAL);
      buttonLayout.setGravity(Gravity.CENTER);

      ImageButton speakerButton = new ImageButton(this);
      LinearLayout.LayoutParams speakerButtonParams = new LinearLayout.LayoutParams(
        100, 100);
      speakerButtonParams.setMargins(8, 0, 8, 0);
      speakerButton.setLayoutParams(speakerButtonParams);
      speakerButton.setImageResource(R.drawable.ic_speaker);
      speakerButton.setScaleType(ImageView.ScaleType.CENTER_INSIDE);
      speakerButton.setBackgroundResource(R.drawable.button_background);
      speakerButton.setPadding(16, 16, 16, 16);
      speakerButton.setOnClickListener(v -> {
```

```
        String textToSpeak = itemName + " was last updated to be at " + location;
            speakText(textToSpeak);
        });

        ImageButton deleteButton = new ImageButton(this);
        deleteButton.setLayoutParams(speakerButtonParams);
        deleteButton.setImageResource(R.drawable.ic_trash_can);
        deleteButton.setScaleType(ImageView.ScaleType.CENTER_INSIDE);
        deleteButton.setBackgroundResource(R.drawable.button_background);
        deleteButton.setPadding(16, 16, 16, 16);
        deleteButton.setOnClickListener(v -> showDeleteConfirmation(position));

        buttonLayout.addView(speakerButton);
        buttonLayout.addView(deleteButton);

        itemLayout.addView(textViewItemName);


      itemLayout.addView(textViewLocation);
       itemLayout.addView(buttonLayout);

        linearItemContainer.addView(itemLayout);
      }
    }


    private void showDeleteConfirmation(final int position) {


AlertDialog.Builder builder;
      builder = new AlertDialog.Builder(this, android.R.style.Theme_Material_Dialog_Alert);
      builder.setTitle("Delete Item")
            .setMessage("Are you sure you want to delete this item?")
            .setPositiveButton(android.R.string.yes, (dialog, which) -> {
              deleteItem(position);
              Toast.makeText(ViewItemsActivity.this, "Item deleted",
      Toast.LENGTH_SHORT).show();
            })
            .setNegativeButton(android.R.string.no, (dialog, which) -> {
              // do nothing
            })
            .setIcon(android.R.drawable.ic_dialog_alert)
            .show();
    }

    private void deleteItem(int position) {
      itemNames.remove(position);
      locations.remove(position);
```

```
        saveItems();
            displayItems();
        }

        private void showDeleteAllConfirmation() {
            AlertDialog.Builder builder;
            builder = new AlertDialog.Builder(this, android.R.style.Theme_Material_Dialog_Alert);



        builder.setTitle("Delete All Items")
                .setMessage("Are you sure you want to delete all items?")
                .setPositiveButton(android.R.string.yes, (dialog, which) -> {
                    deleteAllItems();
                    Toast.makeText(ViewItemsActivity.this, "All items deleted",
            Toast.LENGTH_SHORT).show();
                })
                .setNegativeButton(android.R.string.no, (dialog, which) -> {
                    // do nothing
                })
                .setIcon(android.R.drawable.ic_dialog_alert)
                .show();
        }
}
```

4.2.4 SPLASH PAGE

```
        //SplashActivity.java
        package com.example.reminded;

        import android.content.Intent;
        import android.os.Bundle;
        import android.os.Handler;

        import androidx.appcompat.app.AppCompatActivity;

        public class SplashActivity extends AppCompatActivity {

            // Splash screen timer duration in milliseconds
            private static final int SPLASH_SCREEN_TIMEOUT = 2000;

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

```
// Delayed execution of the MainActivity
        new Handler().postDelayed(() -> {
            // Start the MainActivity
            Intent = new Intent(SplashActivity.this, MainActivity.class);
            startActivity(intent);

            // Close the SplashActivity
            finish();
        }, SPLASH_SCREEN_TIMEOUT);
    }
}
```

4.2.5 STATUS PAGE

```
//statusloc.java
package com.example.reminded;

import android.Manifest;
import android.content.Context;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import java.text.DecimalFormat;

public class statusloc extends AppCompatActivity implements LocationListener {

    private static final int LOCATION_PERMISSION_REQUEST_CODE = 1001;
    private static final String PREF_KEY_HOME_LOCATION_LAT = "home_location_lat";
    private static final String PREF_KEY_HOME_LOCATION_LNG = "home_location_lng";

    private LocationManager;
    private Location homeLocation;
    private TextView textHomeLocation;
    private TextView textCurrentLocation;
    private TextView textStatus;
    private SharedPreferences;
```

```
    private boolean isHomeLocationSet = false;

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

     Button buttonSetHome = findViewById(R.id.buttonSetHome);
     Button buttonDone = findViewById(R.id.buttonDone);



textHomeLocation = findViewById(R.id.textHomeLocation);
     textCurrentLocation = findViewById(R.id.textCurrentLocation);
     textStatus = findViewById(R.id.textStatus);

     locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
     sharedPreferences = getSharedPreferences("com.example.reminded",
     Context.MODE_PRIVATE);

     buttonSetHome.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
           if (checkLocationPermission()) {
              // Request location updates to get the home location
              locationManager.requestSingleUpdate(LocationManager.GPS_PROVIDER,
     statusloc.this, null);
           }
        }
     });

     buttonDone.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
           // Navigate back to MainActivity
           finish();
        }
     });

     // Check if home location is already set
     if (sharedPreferences.contains(PREF_KEY_HOME_LOCATION_LAT) &&
     sharedPreferences.contains(PREF_KEY_HOME_LOCATION_LNG)) {
        double homeLat = sharedPreferences.getFloat(PREF_KEY_HOME_LOCATION_LAT,
     0);
        double homeLng = sharedPreferences.getFloat(PREF_KEY_HOME_LOCATION_LNG,
     0);
        homeLocation = new Location(LocationManager.GPS_PROVIDER);
        homeLocation.setLatitude(homeLat);
        homeLocation.setLongitude(homeLng);
```

```
 isHomeLocationSet = true;
        }

        updateLocation();
        updateStatus();

    }

      private void saveHomeLocation() {
        if (homeLocation != null) {
            SharedPreferences.Editor editor = sharedPreferences.edit();
            editor.putFloat(PREF_KEY_HOME_LOCATION_LAT, (float)



   homeLocation.getLatitude());
            editor.putFloat(PREF_KEY_HOME_LOCATION_LNG, (float)
        homeLocation.getLongitude());
            editor.apply();
        }
      }

      private void updateLocation() {
        if (isHomeLocationSet) {
            String homeLocationText = "Home Location: " + formatLocation(homeLocation);
            textHomeLocation.setText(homeLocationText);
        } else {
            textHomeLocation.setText("Home Location: Unknown");
        }

        Location currentLocation = getLastKnownLocation();
        if (currentLocation != null) {
            String currentLocationText = "Current Location: " + formatLocation(currentLocation);
            textCurrentLocation.setText(currentLocationText);
        } else {
            textCurrentLocation.setText("Current Location: Unknown");
        }
      }

      private String formatLocation(Location location) {
        DecimalFormat = new DecimalFormat("#.00");
        double latitude = Double.parseDouble(decimalFormat.format(location.getLatitude()));
        double longitude = Double.parseDouble(decimalFormat.format(location.getLongitude()));
        return latitude + ", " + longitude;
      }

      private void updateStatus() {
        if (!isHomeLocationSet) {
            textStatus.setText("Set home first");
```

```
} else if (homeLocation == null) {
        textStatus.setText("Unable to retrieve home location");
    } else {
        Location currentLocation = getLastKnownLocation();
        if (currentLocation != null) {
            float distance = currentLocation.distanceTo(homeLocation);
            if (distance < 500) {
                textStatus.setText("Your vehicles are safe");
            } else {
                textStatus.setText("You have locked your house");
            }

        } else {
            textStatus.setText("Unable to retrieve current location");

}
    }
  }
```

4.2.6 NOTIFICATION BAR

```
//MyApplication.java
package com.example.reminded;
import android.app.Activity;
import android.app.Application;
import android.app.Notification;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Intent;
import android.os.Build;
import android.os.Bundle;

public class MyApplication extends Application implements
      Application.ActivityLifecycleCallbacks {

   public static final int NOTIFICATION_ID = 1;




 private static final String CHANNEL_ID = "RemindedChannel";
   private int activityCount = 0;
   private boolean isAppInForeground = false;

   @Override
   public void onCreate() {
```

```
super.onCreate();

        registerActivityLifecycleCallbacks(this);
    }

    @Override
    public void onActivityCreated(Activity activity, Bundle savedInstanceState) {
    }

    @Override
    public void onActivityStarted(Activity activity) {
        if (activityCount == 0) {
            hideNotification();
        }
        activityCount++;
    }

    @Override
    public void onActivityResumed(Activity activity) {
        isAppInForeground = true;
        hideNotification();
    }

    @Override
    public void onActivityPaused(Activity activity) {
        isAppInForeground = false;
    }

    @Override
    public void onActivityStopped(Activity activity) {
        activityCount--;
        if (activityCount == 0 && !isAppInForeground) {
            showNotification();
        }
    }

    @Override
    public void onActivitySaveInstanceState(Activity activity, Bundle outState) {
    }

    @Override
    public void onActivityDestroyed(Activity activity) {
    }

    private void showNotification() {
        createNotificationChannel();

        Intent addIntent = new Intent(this, AddItemsActivity.class);
        addIntent.putExtra("dismiss_notification", true);
```

```
        Intent viewIntent = new Intent(this, ViewItemsActivity.class);

        viewIntent.putExtra("dismiss_notification", true);

        PendingIntent addPendingIntent = PendingIntent.getActivity(
                this, 0, addIntent, PendingIntent.FLAG_UPDATE_CURRENT);
        PendingIntent viewPendingIntent = PendingIntent.getActivity(
                this, 0, viewIntent, PendingIntent.FLAG_UPDATE_CURRENT);

        Notification.Action addAction = new Notification.Action.Builder(
                android.R.drawable.ic_menu_add, "Add", addPendingIntent)
                .build();

        Notification.Action viewAction = new Notification.Action.Builder(
                android.R.drawable.ic_menu_search, "Search", viewPendingIntent)
                .build();

        Notification.Builder builder = new Notification.Builder(this, CHANNEL_ID)
                .setSmallIcon(R.drawable.notification_icon)
                .setContentTitle("Reminded")
                .setContentText("Tap to open the app")
                .addAction(addAction)
                .addAction(viewAction)
                .setDeleteIntent(getDismissIntent());

        NotificationManager =
                (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
        notificationManager.notify(NOTIFICATION_ID, builder.build());
    }

    private void hideNotification() {
        NotificationManager =
                (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
        notificationManager.cancel(NOTIFICATION_ID);
    }

    private void createNotificationChannel() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    CharSequence name = "Reminded Channel";
            String description = "Channel for Reminded notifications";
            int importance = NotificationManager.IMPORTANCE_DEFAULT;
            NotificationChannel channel = new NotificationChannel(CHANNEL_ID, name,
        importance);
            channel.setDescription(description);

            NotificationManager =
                    getSystemService(NotificationManager.class);
            notificationManager.createNotificationChannel(channel);
        }
    }
```

```
    private PendingIntent getDismissIntent() {
        Intent = new Intent(this, DismissNotificationReceiver.class);

        return PendingIntent.getBroadcast(
                this, 0, intent, PendingIntent.FLAG_CANCEL_CURRENT |
        PendingIntent.FLAG_IMMUTABLE);
    }
}
```

4.2.7 DISMISS NOTIFICATION RECEIVER

```
//DismissNotificationReceiver.java
package com.example.reminded;
public class DismissNotificationReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        // Cancel the notification when the broadcast is received
        NotificationManager =
                (NotificationManager)
context.getSystemService(Context.NOTIFICATION_SERVICE);
        notificationManager.cancel(NOTIFICATION_ID);
    }
}
```

CHAPTER 5

RESULTS

5.1 SPLASH SCREEN

A splash screen (Fig 5.1.1) is a screen of the software that displays while the application or other item is loading. After the load is complete, the user is generally taken to another functional screen. Android provides a default splash screen for all projects however a custom splash screen can be created to accommodate animations and required introduction to pages or activities that can provide detailed understanding and engage audiences at the same time towards the content swap. This is not recommended for Android SDK 11 and below; i.e. Android 6 and below. This splash screen custom animation was made with an illustrated image to make a bried adaptation to the color theme which agrees with UI/UX design philosophy.

Fig 5.1.1 Splash page

5.2 HOME SCREEN

On home page user will see three buttons add items,view items,and status button now user can select the buttons for adding items or viewing items or current location after selecting status button .

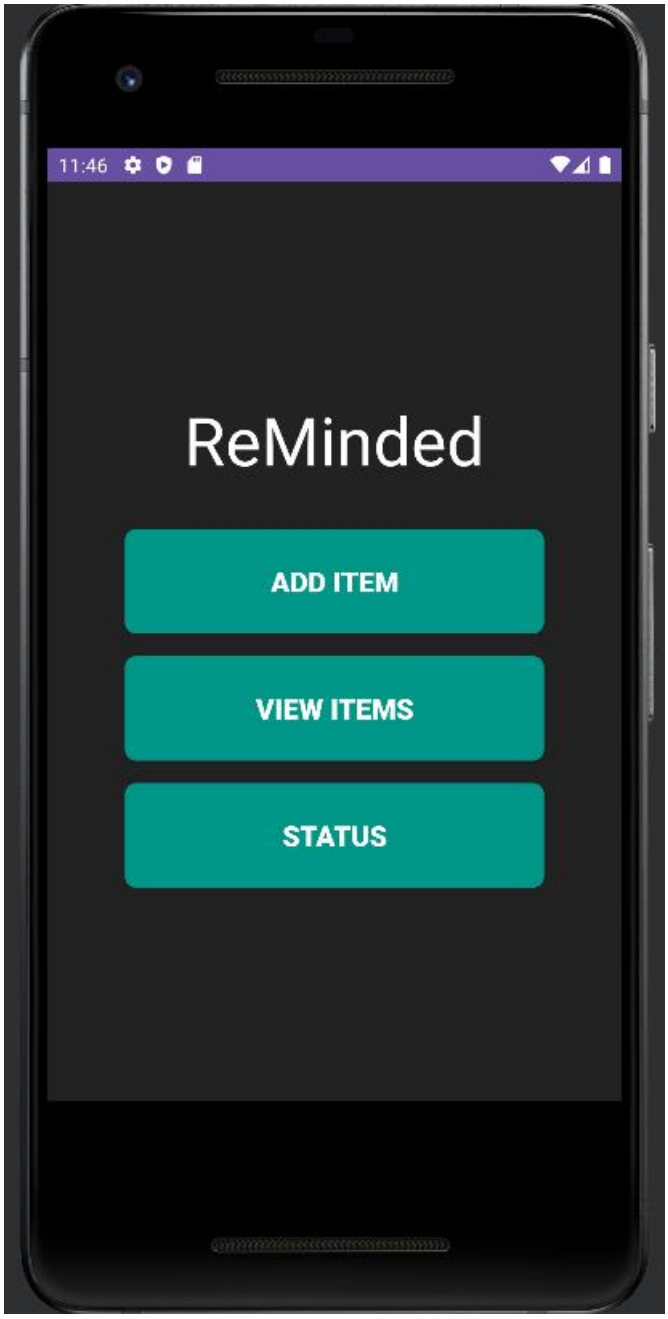

Figure 5.2.1 Home Page

5.3 ADD ITEMS SCREEN

Now user can add items and they can add location for that items where they kept or placed so they can see it later when they press done button they will reach redirect to home page.

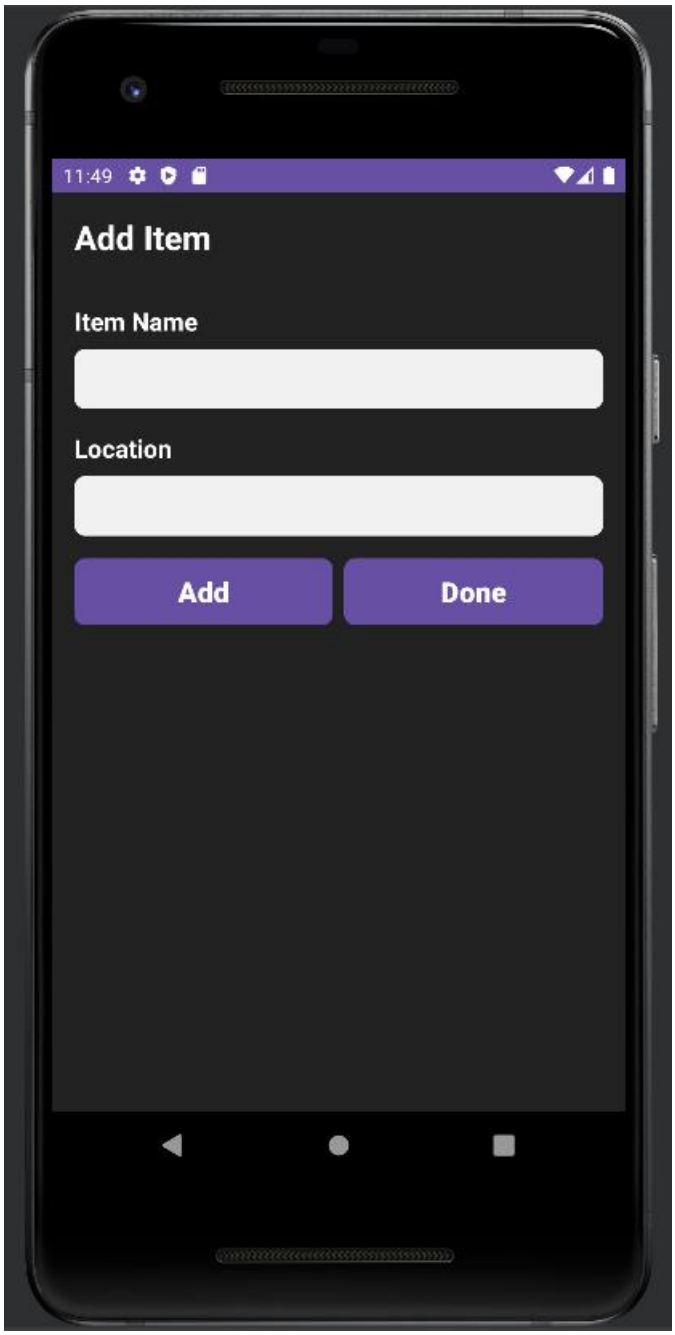

Figure 5.3.1 Add Items Page

5.4 VIEW ITEMS SCREEN

On this page user can see the added items and here they can access items locations by four functionalities text to speech or speech to text or manually or by searching the items now when user will try to search if the item is available it will show but if item is not available anything will not appear but when they press refresh button it will refresh the page and it will show all items again and when they delete symbol on every item it will delete that item or if they press delete all button it will delete every items and last is if they press done button they will reach directly to home page.

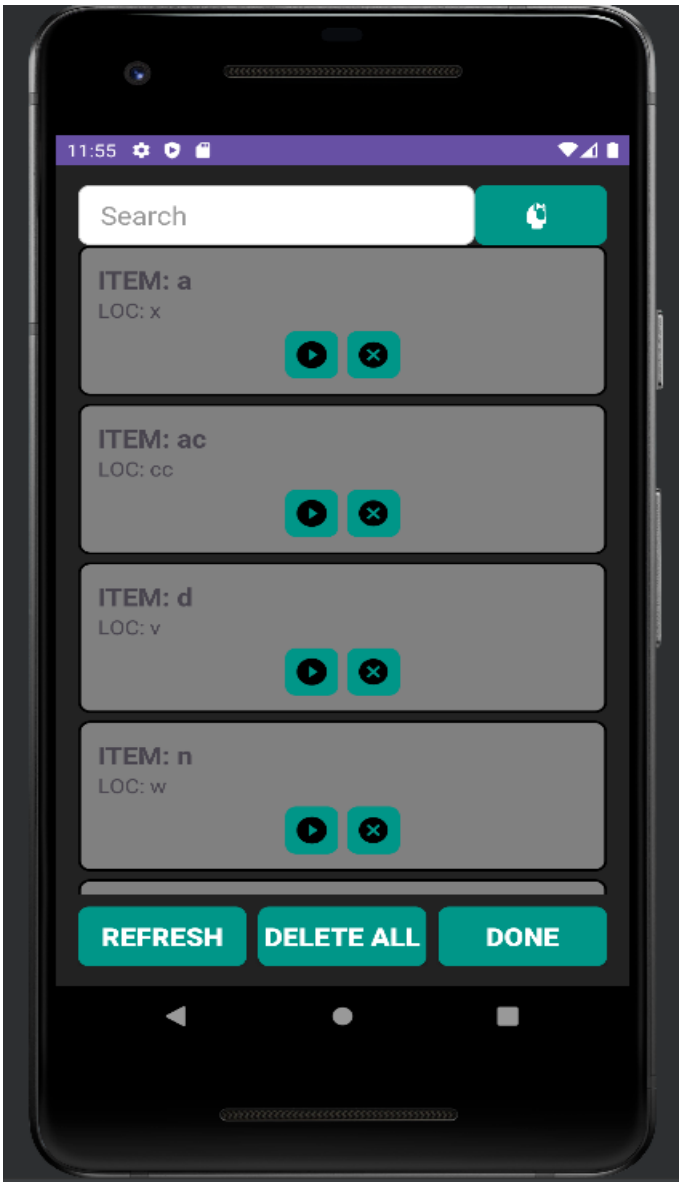

Figure 5.4.1 View Items Page

5.5 STATUS SCREEN

On this page user can set the home location and after setting home location it they go outside they will get message your home is safe and same way if they go somewhere by any vechicle and after coming out of vechile it will again access the location and it will show your vehicle is safe and when they press done they will reach directly to home page.

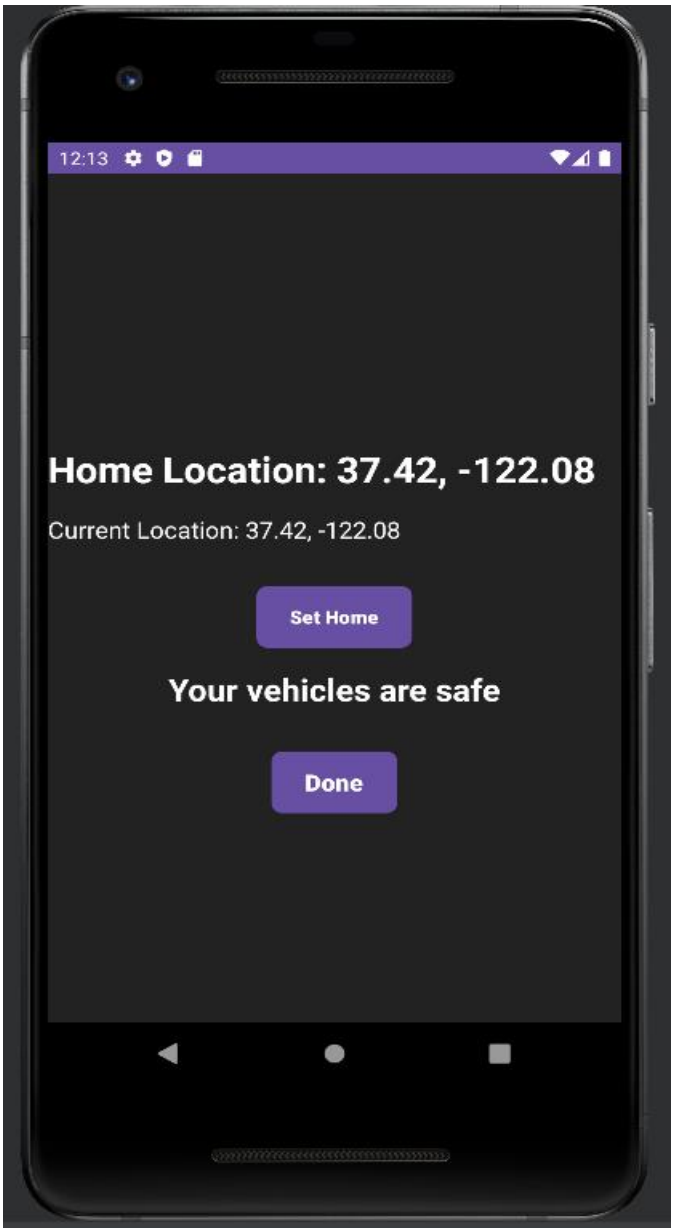

Figure 5.5.1 Status Page

5.6 NOTIFICATION

In this bar user can see the notification for what they put reminder and when they press add it will redirect to add page and when they press search for that reminded item it will go directly to view page.

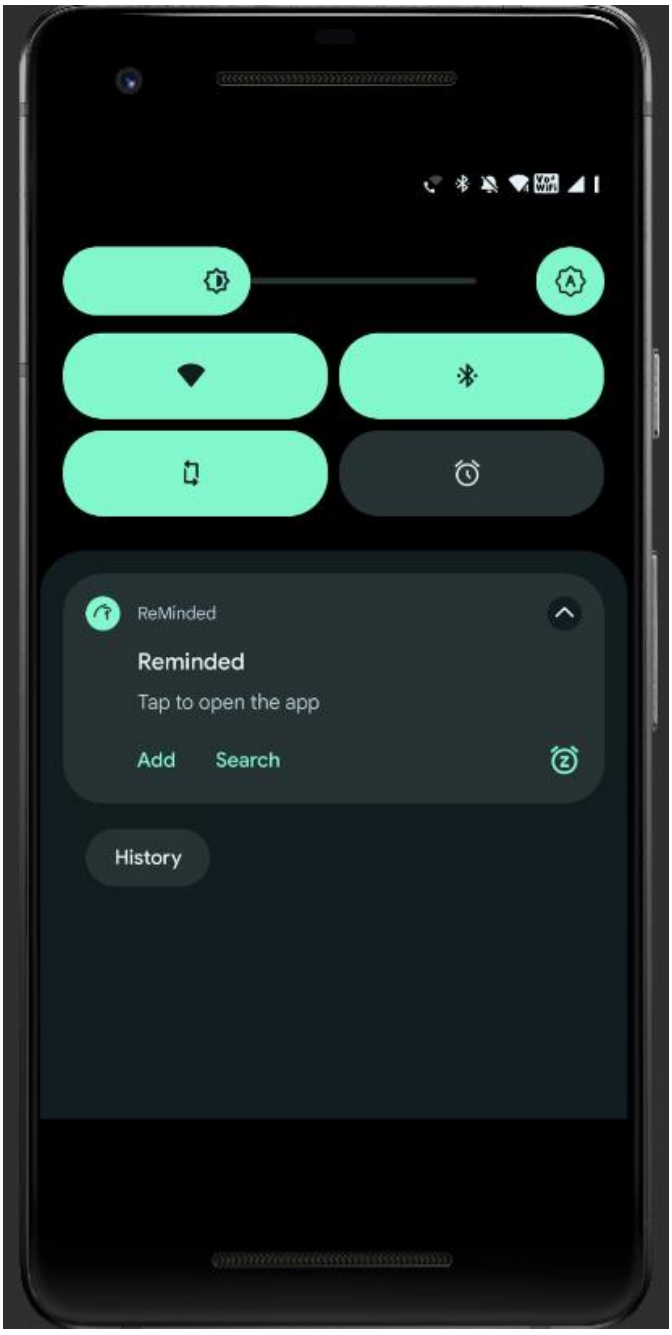

Figure 5.6.1 Notification Bar

CHAPTER 6

CONCLUSION

The Reminded project demonstrates the power of intelligent algorithms and their impact on enhancing the user experience in item management and retrieval. By combining efficient data storage, dynamic user interface generation, search functionality, text-to-speech integration, and speech recognition capabilities, the Reminded Android application provides users with a comprehensive and user-friendly solution.

The algorithm's ability to persistently store item data using the SharedPreferences mechanism ensures that users can seamlessly manage their items across multiple sessions. This reliability and data integrity contribute to a smooth and hassle-free user experience.

The dynamic population of the user interface based on the stored item data enables users to visually organize and navigate their items effectively. The custom layouts and visually appealing design contribute to an intuitive and engaging interface.

The search functionality plays a vital role in helping users quickly locate specific items or groups of items. By filtering the items based on user input, the algorithm simplifies the retrieval process and saves users valuable time and effort.

The integration of text-to-speech capabilities adds an accessibility dimension to the application, allowing users with visual impairments or those who prefer auditory information to interact with the app effectively. The algorithm converts item details into speech, providing users with an alternative means of receiving information.

Furthermore, the speech recognition feature enables users to perform item searches effortlessly by simply speaking their queries. This hands-free approach adds convenience and enhances the user experience, especially in situations where manual input is challenging or not feasible.

The Reminded project, with its intelligent algorithm, offers a practical and user-centric solution for item management and retrieval. By combining advanced functionalities, accessibility features, and a user-friendly interface, it empowers users to efficiently organize, locate, and interact with their items. Overall, the Reminded project showcases the potential of algorithms to simplify and enhance everyday tasks, making the user's life more convenient and productive.

REFERENCES

1. Android Studio Documentation (https://developer.android.com/reference)
2. Vector Image modification (https://developer.android.com/develop/ui/views/graphics/vector-drawable-resources)