

# AIYA!

## 50.001 Introduction to Information Systems & Programming

### Group F04D 1D Project Report

Darren Chan Yu Hao (1006340) | Isaac Koh Jun Wei (1005998)

Lee Le Xuan (1006029) | Michelle Chrisalyn Djunaidi (1006240)

Tay Yang Chun Lucas (1006538) | Wang Jun Long Ryan (1005923)

Brian Lim Yong Jenq (1006386)

# 1. Background

## 1.1. Problem

Existing solutions for reporting of environmental hazards such as pipe leakages, potholes and cluttered litter are not user-friendly and floods the user with too much info. The application made by NEA known as myENV requires a user to navigate to the “report an issue” tab which then shows the users every possible issue that they can report (figure 1), leading to information overload, discouraging users from reporting their issues. Additionally, although a map exists in the application, it only reflects the atmospheric information and is not used to reflect any issues reported by users so other users are unable to be notified of any existing issues. Finally the application is only used to report NEA related issues and not all issues that someone might encounter in their neighbourhood.

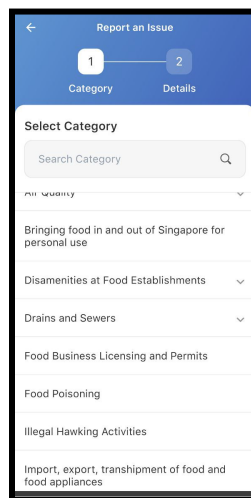


Figure 1: myEnv App

## 1.2. Proposed Solution

Aiya is an Android Development mobile application that enables users to report environmental issues they encounter in their daily lives and raise awareness about them, while also being informed about hazardous areas to avoid.

The app's target audience is the general populace and pedestrians. Inspired by Waze's community based reporting feature that reflects road issues on a real-time map, Aiya!'s real-time map will be updated with custom markers to provide users with information on environmental hazards in their surroundings, along with detailed posts and photos that they can view

Additionally, by posting on Aiya!, the respective councils such as PUB, NEA, SFA will be able to see what issues are affecting the community and resolve them.

Finally, Aiya's user interface is designed to reduce the tedious process of filling out reports by making the ui more visually attractive, through the use of logos and symbols to represent the environmental issues as exemplified in figure 2.

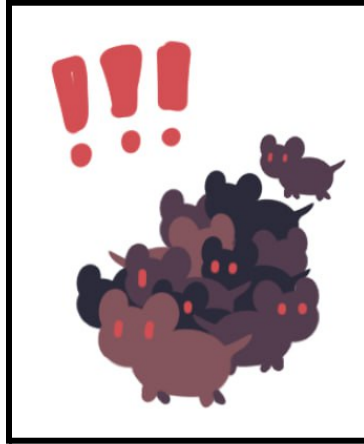


Figure 2: Logo for Rat Infestation

### **1.3. Resources Used**

#### **1.3.1. Google Firebase**

##### **1.3.1.1. Realtime Database**

Realtime Database provides a flexible and scalable data storage solution for Aiya app to build real-time applications that allows for collaboration and synchronisation between multiple users and devices. This means that Aiya can store data like accounts, incident tags and logs in JSON format and allows for any changes made to be immediately propagated to all clients that are granted access.

Refer to Appenix 1.1 for the firebase example.

##### **1.3.1.2. Storage**

Storage provides a secure and scalable solution for storing and serving user-generated content with features such as access control and secure downloads. This allows for Aiya app to store and serve images, and can easily upload and download files directly from devices without the need for any server-side code.

This is vital as Aiya is a community-centric app where all of the content is user-generated; all of the hazard descriptions and photos are uploaded by users.

Refer to Appenix 1.2 for the firebase example.

#### 1.3.1.3. Authentication

Authentication provides Aiya with a secure way to register and authenticate user accounts based on a variety of methods. We have selected to go with email verification. This allows registered users to be logged in to their saved accounts without having to sign up everytime they choose to log in.

Refer to Appenix 1.3 for the firebase example.

#### 1.3.2. Google Maps API

Google Maps API enables the integration of various Google Map features such as maps, direction, and geolocation services into the Aiya app. This means that Aiya app can create customised visual markers that are location-aware on the Singapore city map.

#### 1.3.3. Google Places API

Place API allows us to be able to allow user to key in location details that make sense and we are then able to check and map the location that the user key in into the google maps instead of the user keying random and perhaps non-existant locations

#### 1.3.4. Android Studio IDE

Android Studio IDE provides a set of tools and libraries that allow the Aiya app to be built using Java.

#### 1.3.5. Software components used for the application

Various components taught in the course was used for this application. They are Recycler View, using of intents, executors and other simple components like buttons, textView, editText, cardView and more.

## 2. System Design and Implementation

### 2.1. System Architecture

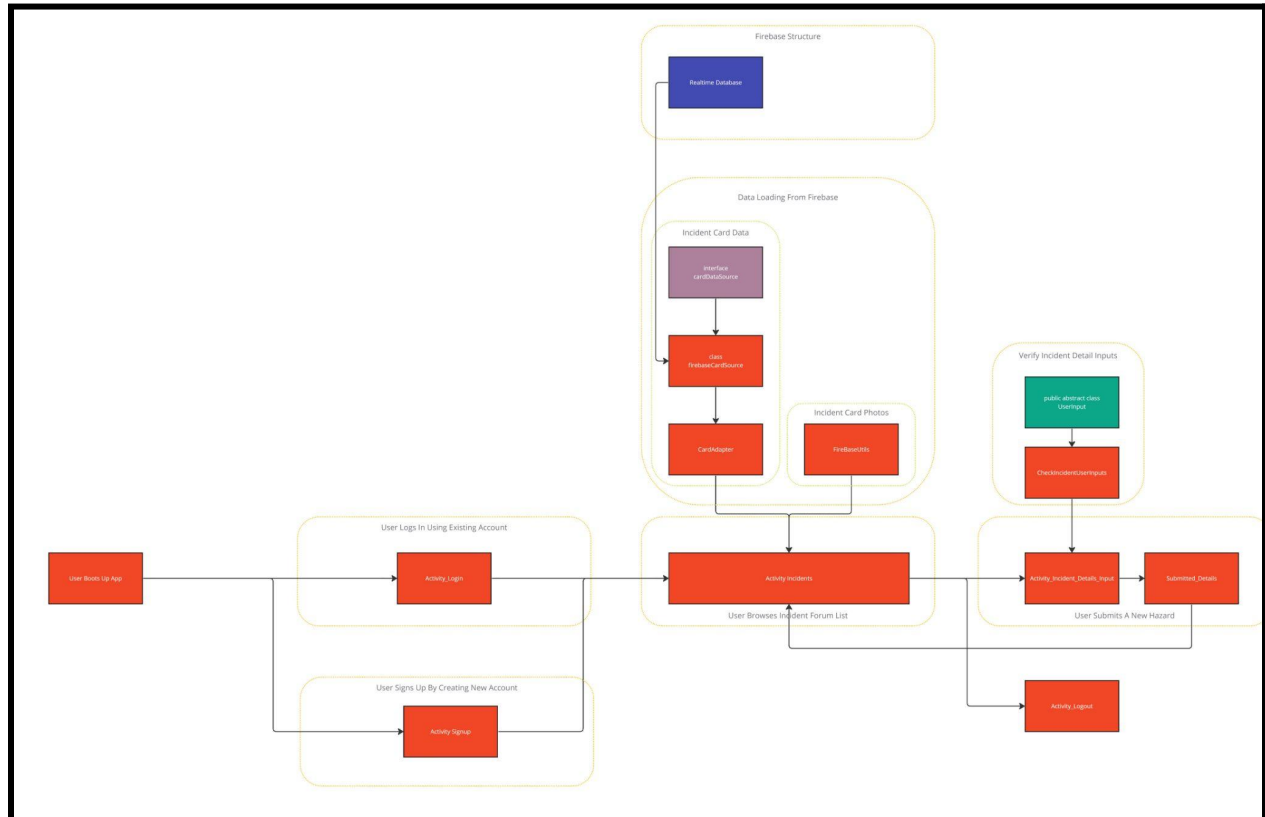


Figure 3: System Architecture  
High quality image attached with submission

## 2.2. App Screenshots/UI

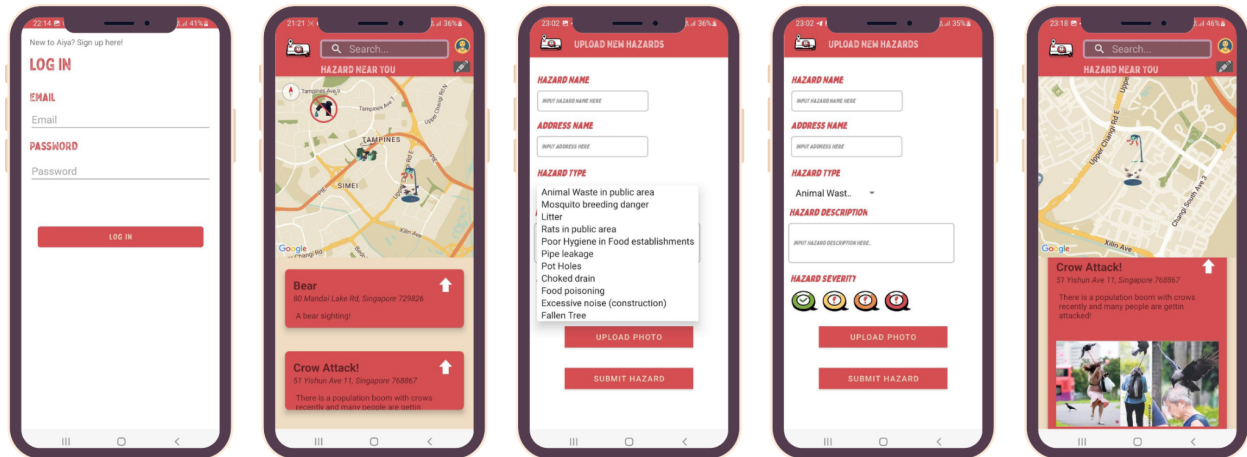


Figure 4: App Ui

## 2.3. Hardware Used

No hardware components were used in the Aiya app.

## 2.4. Design Patterns

### 2.4.1. Singleton Design Pattern

Used in IncidentLog.java to track the incidents and ensure accountability for all incidents. Refer to Appendix 2 for the exact code usage.

### 2.4.2. Adapter Design Pattern

Used in CardAdapter.java to allow for firebase data to be used in the recycler view. Refer to Appendix 3 for the exact code usage.

### 2.4.3. Delegation Design Pattern

Used in Activity\_Incident\_Details\_Input.java to delegate the task of threading with executor and handler to a separate object that implements the template design pattern.

Refer to Appendix 4 for the code examples.

#### 2.4.4. Template Design Pattern

A template for threads to run background and ui tasks will be called by other methods instead of calling executor and handler every time.

This design pattern is also used in `UserInputAbstract.java`, `LoginUserInput.java`, and `SignUpUserInput.java` to check user input details for email and password during Login and Sign Up.

Refer to Appendix 5 for the code usage examples.

### 2.5. Non-Java Usages and Explanation

No non-Java usages were used in the Aiya app

### 3. Responsibilities

Members	Backend Programming	UI Design and Programming	Report
Darren Chan	Incidents	Incidents	Yes
Isaac Koh	Login, Sign Up	Login	Yes
Michelle Chrisalyn Djunaidi	Incidents	Incidents	Yes
Wang Jun Long Ryan	Login, Sign Up	Login	Yes
Lee Le Xuan	Incidents	Incidents	Yes
Lucas Tay	Incidents	Incidents	Yes
Brian Lim	Test case	Test case	Yes



## 4. **Possible Future Work**

Possible future implementation includes the creation of super accounts such as an account that NEA can use to remove posts of issues they have resolved. These accounts can be implemented using the builder design pattern where super accounts have additional attributes that allow them to edit posts.

Another future implementation is the usage of a spam filter to prevent the same posts from being posted. This can be done by comparing the location of posts and matching the type of incident as examples of cases that may be matched. A filter to flag inappropriate images can also be implemented to flag them before posting.

## 5. **Conclusion**

Overall, the Aiya! app achieved the basic goals that were set. The app is able to accurately place pins on a map to indicate the type of hazard and keep track of the users that post using an account system. The app is also able to communicate back and forth with the Firebase to retrieve and push data.

Note: In order for the app to run on android studio, the following line must be added to the local computer's Gradle Script-local.properties:



**MAPS\_API\_KEY=AlzaSyDnIVT6BNvi2ANiKaoYhmteP3WaSGjbuOI**














## Appendix:

Atya-Info-Sys ▾

### Authentication

Users Sign-in method Templates Usage Settings Extensions **NEW**

Add user  

Identifier	Providers	Created ↓	Signed in	User UID
damenfinal@gmail.com		Apr 24, 2023	Apr 24, 2023	GK9TNeirH4TWT2wEbVZJwXMY...
test2@nes.gov.sg		Apr 23, 2023	Apr 23, 2023	vAe4LSQpPaf6GZm8yDY9F93
a@gmail.com		Apr 22, 2023	Apr 22, 2023	K3MDkVUPhleWUw033P5M69Gz...
damen1111111@gmail.com		Apr 14, 2023	Apr 14, 2023	eqyeZi6GnJMK5gle57Qj5Sk11oW2
dhdh@dhdj.com		Apr 14, 2023	Apr 14, 2023	kuL0zUBT9ZVjy6AL7ZvNNtJw4E03
livehacks@gmail.com		Apr 14, 2023	Apr 14, 2023	b448wTbOPbvy79CFbFV9RGA9XL...
a@a.cok		Apr 14, 2023	Apr 14, 2023	sk8kucA.jyoS2uxQCUnbzJslBL63
noesno@tetef.cok		Apr 14, 2023	Apr 14, 2023	u7PhT364Ep0ykcnc2T0HXXCmC2
hellom@gmail.com		Apr 14, 2023	Apr 14, 2023	gg_K9UJxkKeeQrPyQJmQX8PR5...
damen@gmail.com		Apr 14, 2023	Apr 14, 2023	8PUJHu4MSZcDH8hd4g4aLfvM...
damen@damen.com		Apr 14, 2023	Apr 14, 2023	ppC7467Yz85qDL7vhcR0FYnc2
damenishot@damen.com		Apr 14, 2023	Apr 14, 2023	q7y8Qz8Tzbqc2bew168Wt1mFh1
...		...	...	...

Appendix 1.1: Google Firebase Authentication

Alya-info-Sys

## Storage

Files Rules Usage Extensions

gs://alya-info-sys.appspot.com > User Uploaded ...

Upload file

Name	Size	Type	Last modified
1660252234016.jpg	164.48 KiB	image/jpeg	Mar 31, 2023
1660252345801.jpg	129.45 KiB	image/png	Mar 31, 2023
1660252433075.jpg	1.08 MiB	image/jpeg	Mar 31, 2023
1660252461783.jpg	321.42 KiB	image/png	Mar 31, 2023
1660978549118.jpg	141.97 KiB	image/jpeg	Apr 9, 2023
1660978733553.jpg	142.25 KiB	image/jpeg	Apr 9, 2023
1660978911879.jpg	142.25 KiB	image/jpeg	Apr 9, 2023
1660979026017.jpg	142.25 KiB	image/jpeg	Apr 9, 2023
1660979071057.jpg	142.25 KiB	image/jpeg	Apr 9, 2023
1660979157325.jpg	142.25 KiB	image/jpeg	Apr 9, 2023

Appendix 1.2: Google Firebase Storage Database

Alya-info-Sys

## Realtime Database

Data Rules Backups Usage Extensions

https://alya-info-sys-default-rtdb.asia-southeast1.firebaseio.com

Accounts

- Incident Objects
  - NSxtO4J6--e7JjEhKfVr
    - Incident
      - hazardAddress\_Input: "Changi South Avenue 1 (Changi South Ave 1, Singapore)"
      - hazardAddress\_Lat: 1.336502
      - hazardAddress\_Long: 103.9631211
      - hazardDescription\_Input: "Dog litter"
      - hazardImage\_Input: "/User Uploaded Photos Test/1661444274383.jpg"
      - hazardName\_Input: "poopy"
      - hazardType\_Input: "Animal Waste in public area"
      - upvotes: 0
  - NSxxG0PxbNtY2\_n19b
  - NSxx8GaeWgYbqjFKj
  - NSy--grc9eC0u52101e
  - NSy-cJaPDeeFu7W9J2G

Database location: Singapore (asia-southeast1)

Appendix 1.3: Google Firebase Real Time Database

```

package com.example.aiya_test_3.incidents;

public class IncidentLog {
    /*DESCRIPTION
    * The purpose of the IncidentLog is to record all status changes to an incident.
    * Eg: Incident creation, Incident resolution or updates (by NEA).
    *
    * This class applies the singleton (creational) design pattern by creating only on instance
    * of the IncidentLog locally. The reason for this pattern is to ensure that the statuses of
    * incidents are changed in a linear fashion.
    *
    * (Yet to be implemented) In the android lifecycle, onDestroy() should upload the incident log
    * to the database for documentation.
    *
    * METHODS:
    * INFO: records down a status change (upload, update or resolution etc) in the incidents.
    * returnIncidents: return the entire log (should be called onDestroy of the android lifecycle)
    */

    private static IncidentLog instance = null;
    private String log = "";

    // private constructor
    private IncidentLog() {
    }

    public static IncidentLog getInstance() {
        // check if there are any instances of logfile
        if (instance == null) {
            // create an instance of incident log file
            instance = new IncidentLog();
            // table label formatting
            // |Date |Activity Name |Location | but each segment has 15 spaces.
            instance.INFO(String.format("%-12s", "Date") + "|" + String.format("%-15s", "Activity Name")
+ "|" + String.format("%-15s", "Location") + "|");
        }
        return instance;
    }

    // record log
    public void INFO(String msg) {
        log += "\n" + msg; // log activities down.
    }

    public String returnIncidents(){
        return log;
    }
}

```

Appendix 2: Singleton Design Pattern in IncidentLog

```

public class CardAdapter extends RecyclerView.Adapter<CardAdapter.CardViewHolder> {
    private String LastClickPosition;

    LayoutInflater mInflater;
    Context context;
    firebaseCardSource CardData;

    //Firebase Code
    DatabaseReference nRootDatabaseRef;
    DatabaseReference nNodeRef;
    public int positionL = 0;
    private boolean CardClicked;

    private Double hazardLat;
    private Double hazardLng;
    private LatLng hazardLatLng;

    public CardAdapter(Context context, firebaseCardSource CardData) {...}

    public class CardViewHolder extends RecyclerView.ViewHolder {...}

    private void updateFirebaseUpvoteValue(IncidentObject Incident){...}

    @NonNull
    @Override
    public CardViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int viewType){...}

    @Override
    public void onBindViewHolder(@NonNull CardViewHolder holder, int position) {...}

    @Override
    public int getItemCount() {...}

    int vHeight = 0; // instantiate card height

    public void toggleCard(View view, int original_height, TextView hazardName) {...}

    @Override
    public long getItemId(int position) {...}

    @Override
    public int getItemViewType(int position) {...}

    public void updateLastClickPosition(TextView hazardName){...}

    public void setClickedCardPosition() {...}

    public LatLng getClickedCardPosition() {...}

    public boolean isCardClicked() {...}

    public void setCardClicked(boolean clicked) {...}

}

```

Appendix 3: Adapter Design Pattern used in CardAdapter.java

```

class handleBackgroundTask extends BackgroundTasks<DatabaseReference, ArrayList<String>>{

    @Override
    public ArrayList<String> task(DatabaseReference dbInput) {

        ArrayList<String> issues = new ArrayList<String>();
        ArrayList<Boolean> DataRetrievedFlag = new ArrayList<Boolean>();
        DataRetrievedFlag.add(false);

        dbInput.addListenerForSingleValueEvent(new ValueEventListener() {

            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {

                int size = (int) snapshot.getChildrenCount();
                int index = 0;

                for (DataSnapshot dataSnapshot : snapshot.getChildren()) {
                    Object value = dataSnapshot.getValue();
                    if (value != null && value instanceof String) {
                        issues.add(dataSnapshot.getValue(String.class));
                        DataRetrievedFlag.set(0,true);
                    }
                }

            }

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

        });

        while(!DataRetrievedFlag.get(0)) {}

        return issues;
    }

    @Override
    public void done(ArrayList<String> input) {
        if(input != null){
            String[] options = input.toArray(new String[0]);
            // These are the options for the dropdown option

            ArrayAdapter<String> optionAdapter =
                new ArrayAdapter<>(Activity_Incident_Details_Input.this,
                    android.R.layout.simple_spinner_item,
                    options); // Add options to spinner

            HazardTypeDropDownMenu.setAdapter(optionAdapter); // set the options into the drop down
        }
    }
}

```

Appendix 4.1: Using Template and Delegation Design Pattern to do async operation  
(Retrieving Issuelist Drop Down options from firebase)

```

public abstract class BackgroundTasks <I, O> {

    /*****
     * This abstract class uses the template method design pattern. -> to be implemented in the activity.
     * The start method contains all code necessary to run one background task
     * and display its result on an Android activity. (algorithm)
     *
     * The task method handles the background task
     *
     * The done method handles the UI thread task.
     */

    abstract public O task(I input);
    /*****
     * abstract method for the background task.
     * a generic class I, userInput
     * the output of this task, with generic object O will be passed into the method done
     * to run the ui method.
     */

    abstract public void done(O result);
    /*****
     * abstract method for the task on the android UI thread
     * carried out after the background task is completed
     */

    public void start(final I userInput) {
        /*****
         * The start method encompasses the algorithm to perform background and ui thread tasks.
         *
         * Inputs: a generic class I, userInput
         * */

        // UI Task //
        ExecutorsService executor = Executors.newSingleThreadExecutor();
        final Handler handler = new Handler(Looper.getMainLooper());
        executor.execute(new Runnable() {
            @Override
            public void run() {
                // Background Task //
                final Container<O> container = new Container<>();
                O o = task(userInput);
                container.set(o);

                // UI Task - Because of the handler object //
                handler.post(new Runnable() {
                    @Override
                    public void run() {
                        O o = container.get();
                        if( o != null){
                            done(o);
                        }
                    }
                });
            }
        });
    }

    /*****
     * Container class to allow anonymous inner class to read and write to local variables
     * T is a generic object that we can get and set to the container.
     */

    public static class Container<T> {

        T value;

        Container() {
            this.value = null;
        }

        void set(T t) {
            this.value = t;
        }

        T get() {
            return this.value;
        }
    }
}

```

```

package com.example.aiya_test_3.login;

import android.content.Context;
import android.util.Log;
import android.widget.Toast;

import com.google.android.gms.maps.model.LatLng;

public class LoginUserInput extends UserInputAbstract{

    @Override
    public boolean accountInputCheck(String email, String password1, String password2, Context context) {

        // Check if email is valid
        if (!this.checkValidEmail(email)){
            Log.d("LOG IN", "Invalid email attempt");
            Toast.makeText(context, "Please enter valid email", Toast.LENGTH_SHORT).show();
            return false;
        }

        // Check if password is valid
        if (!this.checkValidPassword(password1)){
            Log.d("LOG IN", "Invalid password attempt");
            Toast.makeText(context, "Please enter valid password", Toast.LENGTH_SHORT).show();
            return false;
        }

        // If checks pass, return true
        return true;
    }

    // This is not used in Log In, so we will return null
    @Override
    public String IncidentUserInputCheck(String HazardName,
                                         String HazardAddress,
                                         LatLng HazardAddress_LatLng,
                                         String HazardDescription_Input,
                                         String HazardType_Input,
                                         String HazardImage_Input) {

        return null;
    }
}

```

Appendix 5: Template Design Pattern in LoginUserInput



```

public interface cardDataSource {

    void addWord(String s);
    String getHazardDescription(int i);
    int getSize();

}

import ...;

public class firebaseCardSource implements cardDataSource {

    public static final String FIREBASE_TESTING = "FirebaseTesting";
    DatabaseReference nRootDatabaseRef, nRootDatabaseRef;
    DatabaseReference nNodeRef, nNodeRef;
    Query Post_Order;
    FirebaseStorage storageDatabaseRef;
    StorageReference storageRef;

    int size;
    List<String> hazardDescriptionList, hazardAddressList, hazardNameList, hazardImageList, hazardTypeList;
    List<Double> hazardLatList, hazardLngList;

    List<IncidentObject> IncidentObjectsList;
    String searcher;
    boolean initialDataReadyFlag = false;
    final AtomicInteger numberOfIncident = new AtomicInteger();

    public firebaseCardSource(@Nullable String searcher){...}

    private void countListItems(DataSnapshot snapshot){...}

    private void InitialiseCardItemsList(){...}

    private void setPost_Order(String searcher){...}

    private void repopulateList(DataSnapshot snapshot){...}

    public boolean isInitialDataReadyFlag() {...}

    public long numberOfIncident(){...}

    public String getHazardName(int i){...}

    @Override
    public void addWord(String s) {...}

    public String getHazardDescription(int i){...}

    public String getHazardType(int i) {...}

    public String getHazardAddress(int i){...}

    public LatLng getHazardLatLng(int i) {...}

    public StorageReference getHazardImage(int i) {...}

    public int getSize(){...}

    public IncidentObject getIncidentObject(int i) {...}

    public IncidentObject getIncidentObjectByName(String i) {...}

}

```

Appendix 6: Using of interface for data source