**CHAPTER 1**

**INTRODUCTION**

* 1. **BORDER NAVIGATION AND INCIDENT MANAGEMENT APPLICATION**

Border Management Systems (BMS) applications serve as crucial tools for international border management, offering a range of functionalities to streamline registration, pre-processing, and inspection procedures. These systems facilitate electronic interactions between border agencies and fleet management agencies, enabling seamless coordination and exchange of information. Through the registration and pre-processing modules, BMS applications automate administrative tasks related to border crossings, reducing paperwork and processing times for commercial vehicles and freight containers. Additionally, the border inspection capabilities of these applications enable electronic communications between commercial vehicles, freight containers, and border inspection services, enhancing efficiency and accuracy in border inspections. By digitizing these processes, BMS applications contribute to greater transparency, security, and compliance with international regulations at border checkpoints. Overall, BMS applications play a critical role in modernizing and optimizing border management operations, ensuring smoother and more efficient movement of goods and people across international borders.

**1.2 UTILIZATION OF BORDER SECURITY FORCES AS CENTRALIZED SERVER**

Utilizing border security forces as a centralized server entails repurposing their infrastructure and personnel for data management and dissemination functions akin to a server in an information technology context. This approach capitalizes on the extensive network coverage and surveillance capabilities already established by border security forces. By centralizing data storage and processing within their framework, these forces can efficiently manage and distribute information related to border security, immigration, customs, and other relevant areas. This transformation enhances situational awareness, facilitates real-time decision-making, and streamlines communication channels between different agencies involved in border management. Moreover, leveraging border security forces as a centralized server enhances interoperability and information sharing among national and international stakeholders, thereby bolstering overall border security efforts. This innovative approach optimizes existing resources, strengthens coordination mechanisms, and ultimately fortifies border security infrastructure in an increasingly interconnected world.

**1.3 IMPLEMENTATION OF AUTOMATIC ALERTING SYSTEM FOR EMERGENCY SITUATIONS**

The implementation of an automatic alerting system for emergency situations involves the integration of advanced technology to promptly notify relevant parties when critical events occur. This system employs sensors, data analysis algorithms, and communication channels to detect emergencies such as natural disasters, accidents, or security threats in real-time. Once an emergency is identified, the system automatically triggers alerts through various communication mediums such as text messages, emails, sirens, or mobile applications. These alerts are directed to emergency response teams, relevant authorities, affected populations, and other stakeholders to facilitate swift and coordinated action. Additionally, the system can provide essential information about the nature and location of the emergency, enabling responders to mobilize resources effectively. By automating the alerting process, this system minimizes response times, improves situational awareness, and enhances overall emergency preparedness and response capabilities, ultimately contributing to saving lives and reducing the impact of emergencies on communities.

**1.4 INTEGRATION OF GPS TRACKING TECHNOLOGY FOR MOBILITY MONITORING**

The integration of GPS tracking technology for mobility monitoring involves leveraging Global Positioning System (GPS) technology to track and analyze the movement patterns of individuals, vehicles, or assets in real-time. By equipping objects or individuals with GPS-enabled devices, such as smartphones, vehicle trackers, or wearable gadgets, their precise locations can be continuously monitored and recorded. This data is then transmitted to a centralized system where it can be processed, analyzed, and visualized to provide valuable insights into mobility behavior. This technology finds applications in various domains, including transportation management, logistics, fleet tracking, and personal safety. For instance, in transportation management, GPS tracking enables authorities to monitor traffic flow, optimize routes, and improve overall efficiency. In logistics, it helps track the movement of goods, manage inventory, and enhance supply chain operations. Moreover, GPS tracking technology can be utilized for personal safety purposes, allowing individuals to share their real-time location with trusted contacts or emergency services. Overall, the integration of GPS tracking technology for mobility monitoring offers significant benefits in terms of enhancing operational efficiency, ensuring safety, and enabling informed decision-making across diverse sectors.

**1.5 DEVELOPMENT OF USER-FRIENDLY MOBILE APPLICATION INTERFACE**

The development of a user-friendly mobile application interface focuses on creating an intuitive and seamless interaction experience for users across various devices. This process involves designing the layout, navigation, and visual elements of the application in a way that prioritizes usability and accessibility. User-friendly interfaces typically feature clear and concise menus, easily recognizable icons, and intuitive gestures to navigate through different sections of the app. Additionally, developers pay attention to factors such as font size, color contrast, and interactive elements to ensure that the interface is accessible to users with diverse needs and preferences. Usability testing and feedback loops are integral parts of the development process, allowing developers to identify and address any usability issues or pain points that users may encounter. Ultimately, the goal of developing a user-friendly mobile application interface is to enhance user satisfaction, engagement, and retention by providing a smooth and enjoyable experience that aligns with the expectations and preferences of the target audience.

**1.6 UTILIZATION OF IMEI NUMBER AS UNIQUE IDENTIFIER FOR TRACKING**

The utilization of the International Mobile Equipment Identity (IMEI) number as a unique identifier for tracking provides a valuable tool for various purposes, notably in the realm of mobile device security and management. IMEI numbers are unique identifiers assigned to every mobile device, serving as a digital fingerprint that distinguishes one device from another. By leveraging IMEI numbers, authorities, mobile network operators, and individuals can track and monitor the whereabouts of mobile devices in real-time. This capability proves particularly useful in scenarios such as locating lost or stolen devices, preventing device theft, and enforcing mobile device policies in organizational settings. Additionally, IMEI-based tracking enables law enforcement agencies to trace the movement of devices involved in criminal activities, aiding in investigations and recovery efforts. However, it's crucial to ensure privacy safeguards and regulatory compliance when utilizing IMEI numbers for tracking purposes, as improper use or unauthorized access to this information could infringe on individuals' privacy rights. Overall, the utilization of IMEI numbers as a unique identifier for tracking offers a valuable means of enhancing mobile device security, facilitating device management, and supporting law enforcement efforts while simultaneously necessitating careful consideration of privacy implications and regulatory compliance.

**CHAPTER 2**

**LITERATURE SURVEY**

# 2.1 TITLE: NEAR REAL-TIME TRACKING OF IOT DEVICE USERS

# AUTHOR: Jinseong Kim, Jae J. Jang, [Im Y. Jung](https://ieeexplore.ieee.org/author/37085359829)

**DESCRIPTION:**

This paper presents a straightforward tracking scheme for Internet of Things (IoT) devices. By capturing timed snapshots of IoT devices, the movements and locations of their owners can be indirectly and efficiently traced. To minimize the overhead associated with real-time tracking of mobile IoT devices, the proposed scheme adopts quasi-real-time tracking using the LIDx protocol and its period. It collects only essential information in each snapshot, such as time, remaining energy in the mobile device, and the type of wireless communications interface. Since the users are tracked without authentication, their privacy regarding their mobility, including their identities and associated locations, is preserved. This simplified location tracking method is anticipated to be beneficial in rescuing individuals in perilous situations resulting from disasters or accidents in areas like basements or enclosed spaces such as ships, where GPS coverage may be limited or public communication signals are weak and unstable.

# 2.2 TITLE: A SECURE TRACKING SYSTEM FOR GPS-ENABLED MOBILE PHONES

**AUTHOR:** Hasan Tahsin Bilgic, Ali Ziya Alkar.

**DESCRIPTION:**

This paper presents a secure location tracking system comprising a web interface and client software. Designed to be portable across embedded devices capable of running Java, the system is specifically tailored for use on mobile phones. Given the widespread adoption of mobile phones in today's market, such applications are becoming more prevalent compared to dedicated global positioning systems (GPS). Consequently, we outline the fundamentals of a novel positioning system enhanced with AES security for transmitting GPS data. The encryption key is derived from the IMEI, a unique code assigned to each mobile phone, thus endowing the system with a distinctive feature compared to conventional approaches. Additionally, we introduce geo-fencing to enhance the system's usability, particularly for clients requiring closer tracking.

# 2.3 TITLE: A SECURE TRACKING SYSTEM FOR GPS-ENABLED MOBILE PHONES

**AUTHOR:** Hasan Tahsin Bilgi, Ali Ziya Alkar.

**DESCRIPTION**

This paper presents a secure location tracking system comprising a web interface and client software. It is designed to be compatible with embedded devices running Java, specifically targeting mobile phones. Given the increasing prevalence of mobile phone usage, especially in comparison to dedicated GPS systems, this system aims to cater to this growing demand. The paper outlines the fundamentals of a novel positioning system with enhanced security, achieved through the integration of AES encryption during GPS data transmission. The encryption key is derived from the IMEI, a unique identifier assigned to each mobile phone, thus distinguishing this system from conventional approaches. Additionally, geo-fencing functionality has been incorporated to enhance usability, particularly for clients requiring more precise tracking.

# 2.4 TITLE: APPLICATIONS OF IMEI-BASED TRACKING IN LAW ENFORCEMENT

**AUTHOR: Wang, H., & Chen, L.**

**DESCRIPTION**

The paper by Wang, H., & Chen, L. delves into the application of IMEI-based tracking in law enforcement, offering valuable insights through case studies and best practices. It meticulously examines various instances where IMEI tracking has facilitated successful investigations and operations, underscoring its pivotal role in combating crimes involving mobile devices. Through these case studies, the paper elucidates the effectiveness of IMEI tracking in locating stolen devices, apprehending suspects, and dismantling criminal networks. Moreover, it underscores the significance of collaborative efforts between law enforcement agencies, mobile network operators, and technology providers in leveraging IMEI-based tracking capabilities to their fullest potential. By highlighting real-world examples and best practices, the paper provides law enforcement professionals with practical guidance on optimizing the utilization of IMEI-based tracking tools and enhancing their effectiveness in addressing mobile device-related crimes. Ultimately, this comprehensive analysis underscores the crucial role of IMEI-based tracking in bolstering law enforcement efforts and safeguarding communities from the illicit activities facilitated by mobile devices.

# 2.5 TITLE: UNDERWATER FISH IDENTIFICATION IN REAL-TIME USING CONVOLUTIONAL NEURAL NETWORK

**AUTHOR:** [**Hanrong Lu**](https://ieeexplore.ieee.org/author/37087465266)**;**[**Xin Chen**](https://ieeexplore.ieee.org/author/37087212518)**;**[**Xuhui Lan**](https://ieeexplore.ieee.org/author/37087465238)**;**[**Feng Zheng**](https://ieeexplore.ieee.org/author/37087198632)

**DESCRIPTION:** S.K. Aruna; N. Deepa; T Devi.

Artificial intelligence (AI) encompasses a broad range of applications that learn from data provided and process it in a manner akin to the human brain. When a computer program mimics a characteristic of the human brain, it is considered innovative. Various methods, including statistical approaches and traditional algorithms, are employed to validate the effectiveness of AI. The proliferation of AI is closely tied to the availability of vast storage and copious data, encompassing diverse formats such as text messages, images, videos, and geospatial information. Machine learning, a subset of AI, includes deep learning, which involves multiple layers of neural networks, enabling the system to learn and emulate brain functions. Adding more hidden layers to neural networks enhances efficiency. Deep learning is employed to gather and process data. In the context of aquaculture production, challenges arise in accurately counting fish during spawning. Previous methods relied on manual or automated counting techniques, leading to inaccuracies. To address this, a proposed method combines IoT techniques with image detection to improve accuracy. Image data is categorized based on frequency, with 8200 images used for training and 2500 for verification. Only relevant data sources are utilized during training and verification to refine parameters and optimize the VGG19 model. As a result, the improved model achieves an impressive accuracy of 98%.

**CHAPTER 3**

**SYSTEM ANALYSIS**

* 1. **EXISTING SYSTEM**

The Indian Coast Guard plays a multifaceted role, with one of its key responsibilities being the enhancement of fisherman safety. While the Coast Guard is not solely dedicated to fisherman assistance, ensuring the welfare of fishers remains integral to its objectives. Strategically, the Coast Guard is tasked with safeguarding maritime zones against illicit activities such as illegal infiltration via maritime routes and environmental degradation. Additionally, it is mandated to provide humanitarian aid and support scientific endeavors within the maritime domain. However, the current system presents challenges for fishers, particularly concerning their awareness of maritime borders and the resultant risks they face, often leading to tragic outcomes. From a technical standpoint, addressing these challenges requires the implementation of advanced navigation and communication systems tailored to the needs of fishers. This may involve the integration of GPS technology into fishing vessels to provide real-time location data, ensuring fishers remain within safe and designated maritime zones. Additionally, the development of educational programs and initiatives aimed at increasing fishers' understanding of maritime borders and navigation protocols is crucial. Moreover, leveraging satellite imagery and remote sensing technologies can aid in identifying potential hazards and monitoring maritime activities, enhancing overall safety and security for fishers operating in coastal waters. By combining technical solutions with strategic initiatives, the Indian Coast Guard can effectively mitigate risks and better protect the lives and livelihoods of fishers.

**3.1.2 Disadvantages**

* Fisherman die since they don’t know the border
* Relatives cannot able to know about the fisherman
* Inadequate Communication Infrastructure

**3.2 PROPOSED SYSTEM**

The proposed system aims to mitigate navigation and safety concerns for individuals, particularly Tamil fishermen, operating in maritime border regions. Leveraging advanced technology, the system offers real-time route information to ensure safe navigation. Central to its functionality is the utilization of Indian border security forces' infrastructure as a centralized server for effective coordination and dissemination of notifications. Key features include device-based tracking using GPS technology for precise location data, independent of network availability, and a notification mechanism to alert users and authorities about safety or security issues, such as the presence of opposing forces in ships. While benefiting all border region users, the system is specifically tailored to Tamil fishermen, providing essential navigational assistance and security alerts to enhance their safety and operational efficiency. Technically, the system relies on a robust GPS infrastructure and secure communication protocols to ensure accurate location tracking and data transmission integrity. Overall, the proposed system offers a comprehensive solution to address navigation and safety challenges, leveraging collaboration with border security forces and advanced technology to enhance safety and well-being in maritime environments.

**3.2.1 Advantages of the Proposed System**

* Border can be known to the user
* An alert system can be known to the fisherman when they cross the border
* The intimation will be made to the emergency numbers.

**3.3 SYSTEM ARCHITECTURE**

System architecture can be defined as a structure composed of components, and rules characterizing the interaction of these components. System architecture diagram is a starting point of product description visualization – a product structure overview. Three types of system architectures are identified, integrated, distributed and mixed, (partly integrated and partly distributed). It is shown that the type of interfaces defines the type of architecture. Integrated systems have more interfaces, which furthermore are vaguely defined. An application architecture diagram provides a high-level graphical view of the application architecture, and helps you identify applications, sub-applications, components, databases, services, etc, and their interactions. See Application and Business Services (EAM). System - A packaged application.

If you restricted yourself to four layers, they may be defined as:

1) Algorithm,

2) Programming language/compiler,

3) processor/memory,

4) I/O. Other abstraction definitions may contain three layers:

1) Application,

2) System software,

3) Hardware

**Various organizations define systems architecture in different ways, including:**

* An allocated arrangement of physical elements which provides the design solution for a consumer product or life-cycle process intended to satisfy the requirements of the functional architecture and the requirements baseline.
* Architecture comprises the most important, pervasive, top-level, strategic inventions, decisions, and their associated rationales about the overall structure (i.e., essential elements and their relationships) and associated characteristics and behavior.
* If documented, it may include information such as a detailed inventory of current hardware, software and networking capabilities; a description of long-range plans and priorities for future purchases, and a plan for upgrading and/or replacing dated equipment and software.



Fisher man

Update numbers



Location track



Update location

Set fence

Family members and friends



**Fig 3.1 System Architecture**

**CHAPTER 4**

**SYSTEM REQUIREMENTS**

**4.1 HARDWARE REQUIREMENTS**

* Processor : Dual core processor 2.6.0 GHZ
* RAM : 1GB
* Hard disk : 160 GB
* Compact Disk : 650 Mb
* Keyboard : Standard keyboard
* Monitor : 15 inch color monitor

**4.2 SOFTWARE REQUIREMENTS**

* Front End : Android
* IDE : Eclipse
* Platform : Windows 7

**CHAPTER 5**

**MODULES DESCRIPTION**

**5.1 MODULES**

* **Update number**
* **Set position**
* **Location track**
* **Fisherman Intimation**
* **Emergency Alert**
  1. **MODULES DESCRIPTION**
     1. **Update number**

An Android-based application will be developed and distributed to fishermen and their relatives to provide comprehensive details about the fishermen's activities and whereabouts. The application will include features such as real-time location tracking, vessel status updates, and emergency contact information. Technically, the application will utilize GPS technology to track the fishermen's locations accurately, ensuring precise monitoring even in remote maritime areas. Additionally, the application will allow fishermen to update their status, indicating their current activities and intended routes. Emergency contact numbers, typically 3-4 in number, will be integrated into the application, enabling quick communication with relevant authorities in case of emergencies or border crossings. Furthermore, the application will incorporate secure communication protocols to safeguard sensitive data and ensure the integrity of information transmission. Overall, the Android application will serve as a vital tool for fishermen and their relatives, providing them with essential information and facilitating prompt response in emergency situations.

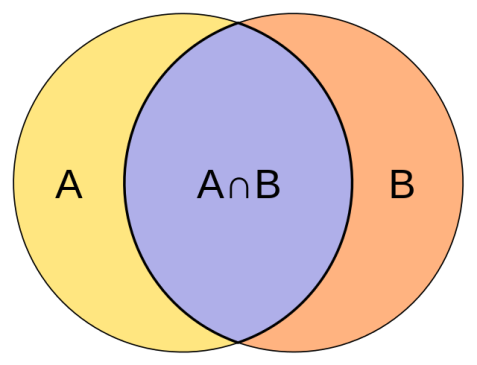
* + 1. **SET POSITION**

The administration side of the proposed system will facilitate the dynamic updating of border positions based on the intended routes to be used by fishermen. Administrators will have the capability to input latitude and longitude values to define the borders, which will then be utilized for monitoring purposes. These borders will typically be set approximately 500 meters from the government-allotted border to ensure a safety margin for fishermen. From a technical perspective, the system will employ geospatial data management techniques to accurately define and update border positions. Geospatial databases will store and manage the latitude and longitude values of these borders, allowing for efficient retrieval and manipulation. Additionally, the system will utilize algorithms for proximity detection to determine when fishermen cross these defined borders. Once a border crossing is detected, the system will trigger notifications to alert relevant authorities and stakeholders. Implementation-wise, the system will incorporate geofencing technology to create virtual boundaries around the defined borders. Geofencing APIs provided by mobile platforms such as Android will be utilized to monitor when fishermen enter or exit these virtual boundaries in real-time. Furthermore, the system will leverage GPS technology to ensure precise location tracking of fishermen relative to the defined borders. Overall, the integration of geospatial data management, geofencing technology, and GPS tracking capabilities will enable effective monitoring and notification of border crossings for enhanced safety and security of fishermen.

**SIMILARITY INDEX ALGORITHM**

The structural similarity index measure (SSIM) is a method for predicting the perceived quality of data, as well as other kinds of digital images and videos. SSIM is used for measuring the similarity between two images. The similarity is measure with the concern Meta data of the digital data extraction. The intersection of the data with the similarity extraction will be identified. This method shows the duplicate data elimination system

SSIM is used as a metric to measure the similarity between two given images. As this technique has been around since 2020, a lot of material exists explaining the theory behind SSIM but very few resources go deep into the details, that too specifically for a gradient-based implementation as SSIM is often used as a loss function.



**Fig 5.2 Intersection of Meta data**

The Structural Similarity Index (SSIM) is a perceptual metric that quantifies image quality degradation caused by processing such as data compression or by losses in data transmission. It is a full reference metric that requires two images from the same image capture a reference image and a processed image. The processed image is typically compressed. It may, for example, be obtained by saving a reference image as a JPEG (at any quality level) then reading it back in. SSIM is best known in the video industry, but has strong applications for still photography.

**5.2.3.1 LOCATION TRACK**

A GPS navigation device functions by receiving signals from satellites within the Global Positioning System (GPS), enabling it to determine the device's precise location on Earth. These signals provide latitude and longitude information, and in some cases, altitude calculations may also be included. GPS devices find applications across various sectors, including military, aviation, marine, and consumer products. In the context of tracking fishermen, the GPS system serves to monitor their location in real-time. Technically, GPS devices use a network of at least 24 satellites orbiting the Earth to triangulate the device's position based on the time it takes for signals to travel from the satellites to the device. This information is then processed by the device's software to calculate the latitude and longitude coordinates. Additionally, GPS devices may employ algorithms to compare the current position with predefined restricted areas. When a fishing boat approaches within a specified distance, typically 500 meters, of a restricted area, the device triggers alerts or notifications to inform relevant authorities or stakeholders. This functionality relies on geofencing technology, which creates virtual boundaries around restricted areas and monitors when a device enters or exits these boundaries. Overall, GPS navigation devices play a crucial role in tracking fishermen and ensuring compliance with regulations or safety protocols in marine environments.

**Global Positioning System**

GPS, or Global Positioning System, is a satellite-based navigation system that provides location and time information anywhere on Earth where there is an unobstructed line of sight to four or more GPS satellites. The system consists of a constellation of at least 24 satellites orbiting the Earth, managed by the United States government's Department of Defense. Technically, GPS works by utilizing a principle called trilateration. Each GPS satellite continuously broadcasts signals containing its precise time and location. A GPS receiver on the ground, such as a GPS navigation device, picks up signals from multiple satellites. By calculating the time it takes for each signal to reach the receiver, along with the precise position of each satellite at the time of transmission, the GPS receiver can determine its own distance from each satellite. Once the receiver has obtained distance measurements from at least four satellites, it can use trilateration to calculate its own three-dimensional position (latitude, longitude, and altitude). Trilateration involves intersecting spheres, with each satellite representing the center of a sphere and the distance to the receiver representing the radius. By finding the point where these spheres intersect, the receiver can pinpoint its location. In addition to providing location information, GPS satellites also broadcast signals indicating the current time, which allows GPS receivers to synchronize their clocks accurately. This precise timekeeping is crucial for various applications, including navigation, timing, and synchronization of telecommunications networks. Overall, GPS technology relies on a network of satellites, precise timing, and complex mathematical algorithms to determine accurate positions on Earth, enabling a wide range of applications in navigation, mapping, surveying, and more.

**Fig 5.3 Global Positioning System**

* + 1. **FISHERMAN INTIMATION**

Location estimation plays a pivotal role in the effective tracking of fishermen, ensuring prompt notification in the event of border crossings. Utilizing geofencing technology, the system establishes virtual boundaries around predefined areas, such as restricted zones. Geofencing involves the use of GPS or RFID technology to create these virtual perimeters, allowing for real-time monitoring of the fishermen's movements. When a fishing vessel breaches these geofenced boundaries, typically positioned approximately 500 meters from restricted areas, the system initiates an automated process to trigger alerts or notifications. These notifications are transmitted through various communication channels, including SMS or push notifications, to pre-registered numbers of relatives or authorities. From a technical perspective, geofencing algorithms continuously monitor the GPS coordinates of the fishing vessel, comparing them to the predefined boundaries to detect any breaches. Upon detection, the system executes predefined actions, such as sending notifications, to ensure timely response and intervention. By leveraging geofencing and notification mechanisms, the system enhances the safety and security of fishermen by enabling swift responses to border crossings or emergency situations.

* + 1. **EMERGENCY ALERT**

An alert system will be implemented to notify fishermen when they have crossed predefined borders, facilitating timely awareness of their location relative to designated boundaries. This notification mechanism is instrumental in assisting fishermen and their relatives in monitoring their position, especially in the absence of clear delineations of borders at sea. Technically, the alert system will utilize geofencing technology, which involves defining virtual boundaries using GPS coordinates. These boundaries will be established around specified areas, such as maritime borders or restricted zones. When a fishing vessel breaches these virtual boundaries, typically set at a distance of 500 meters from the designated border, the system will trigger alerts or notifications. These notifications will be transmitted via SMS, push notifications, or other communication channels to both the fishermen and their registered relatives or authorities. From a technical standpoint, the alert system will employ algorithms to continuously monitor the GPS coordinates of the fishing vessel and compare them to the predefined boundaries. Upon detecting a border crossing, the system will initiate the notification process, ensuring that relevant parties are promptly informed of the situation. By leveraging geofencing technology and notification mechanisms, the system enhances the safety and navigation capabilities of fishermen operating in maritime environments.

**CHAPTER 6**

**SYSTEM TESTING**

**6.1 UNIT TESTING**

The first test in the development process is the unit test. The source code is normally divided into modules, which in turn are divided into smaller units called units. These units have specific behavior. The test done on these units of code is called unit test. Unit test depends upon the language on which the project is developed. Unit tests ensure that each unique path of the project performs accurately to the documented specifications and contains clearly defined inputs and expected results. A Unit corresponds to a screen /form in the package. Unit testing focuses on verification of the corresponding class or Screen. This testing includes testing of control paths, interfaces, local data structures, logical decisions, boundary conditions, and error handling. Unit testing may use Test Drivers, which are control programs to co-ordinate test case inputs and outputs and Test stubs, which replace low-level modules. A stub is a dummy subprogram.

**6.2 INTEGRATION TESTING**

Testing in which modules are combined and tested as a group is known as integration testing. Modules are typically code modules, individual applications, source and destination applications on a network, etc. Integration Testing follows unit testing and precedes system testing. Testing after the product is code complete. Betas are often widely distributed or even distributed to the public at large in hopes that they will buy the final product when it is release. Integration testing is used to verify the combining of the software modules. Integration testing addresses the issues associated with the dual problems of verification and program construction.

**6.3 SYSTEM TESTING**

System testing is a critical aspect of Software Quality Assurance and represents the ultimate review of specification, design and coding. Testing is a process of executing a program with the intent of finding an error. A good test is one that has a probability of finding an as yet undiscovered error. The purpose of testing is to identify and correct bugs in the developed system. Nothing is complete without testing. Testing is the vital to the success of the system. In the code testing the logic of the developed system is tested. For this every module of the program is executed to find an error. To perform specification test, the examination of the specifications stating what the program should do and how it should perform under various conditions.

**6.4 FUNCTIONAL TESTING**

It is a type of software testing whereby the system is tested against the functional requirements/specifications. Functions (or features) are tested by feeding them input and examining the output. Functional testing ensures that the requirements are properly satisfied by the application. This type of testing is not concerned with how processing occurs, but rather, with the results of processing. It simulates actual system usage but does not make any system structure assumptions.

Apart from these tests, there are some special tests conducted which are given below:

**6.5 PERFORMANCE TIME TESTING**

This test determines the length of the time used by the system to process transaction data. In this phase the software developed Testing is exercising the software to uncover errors and ensure the system meets defined requirements. Testing may be done at 4 levels.

* Module Level
* Regression

**6.5.1 Module Level Testing**  
 Module Testing is done using the test cases prepared earlier. Module is defined during the time of design.

**6.5.2 Regression Testing**

Each modification in software impacts unmodified areas, which results serious injuries to that software. So the process of re-testing for rectification of errors due to modification is known as regression testing. Delivery Installation and Delivery are the process of delivering the developed and tested software to the customer. Refer the support procedures Acceptance and Project Closure Acceptance is the part of the project by which the customer accepts the product. This will be done as per the Project Closure, once the customer accepts the product closure of the project is started. This includes metrics collection, PCD etc.

**CHAPTER 7**

**CONCLUSION**

In conclusion, the implementation of a border-crossing alert system for fishermen represents a significant advancement in ensuring their safety and security while navigating maritime environments. By leveraging geofencing technology and notification mechanisms, the system provides timely alerts to fishermen and their relatives or authorities when they cross predefined borders. This enhances situational awareness and enables prompt response in emergency situations. The system addresses the critical need for improved navigation assistance and border monitoring capabilities for fishermen, particularly in areas where maritime borders may not be clearly delineated. Moving forward, future enhancements can further refine and augment the capabilities of the system to better serve the needs of fishermen and enhance their overall safety and well-being.

**CHAPTER 8**

**FUTURE ENHANCEMENTS**

In future enhancements, several avenues exist to further refine and enhance the border-crossing alert system for fishermen. Firstly, advanced navigation features could be integrated, incorporating route optimization algorithms and real-time weather updates to assist fishermen in selecting the safest and most efficient routes. Enhanced communication capabilities, such as satellite-based communication systems and two-way messaging functionality, could also be implemented to facilitate seamless communication between fishermen and authorities in remote maritime areas. Additionally, the incorporation of artificial intelligence (AI) and machine learning (ML) algorithms could enhance the predictive capabilities of the system, enabling it to anticipate potential risks and provide proactive safety recommendations to fishermen. Moreover, the system's geofencing boundaries could be expanded to include additional critical areas for monitoring, such as marine conservation zones or high-risk maritime zones prone to illegal activities. Lastly, integration with emergency response services, such as coast guard agencies or maritime rescue organizations, could enable automated distress signal transmission and coordinated rescue operations in case of emergencies. These future enhancements hold the potential to further elevate the effectiveness and utility of the border-crossing alert system, ultimately enhancing the safety and security of fishermen at sea.

**APPENDIX I**

**SAMPLE SOURCE CODE**

**Home**

package com.example.border\_crossing\_fisherman;

import java.io.DataInputStream;

import java.net.HttpURLConnection;

import java.net.URL;

import android.os.AsyncTask;

import android.os.Bundle;

import android.os.Handler;

import android.app.Activity;

import android.app.Dialog;

import android.app.ProgressDialog;

import android.content.Intent;

import android.util.Log;

import android.view.Menu;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.ImageView;

import android.widget.TextView;

import android.widget.Toast;

public class home extends Activity {

TextView t1;

public static String uemail="";

ProgressDialog pDialog;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_home);

// e1 = (EditText) findViewById(R.id.editText1);

// e2 = (EditText) findViewById(R.id.editText2);

// b1 = (Button) findViewById(R.id.button2);

t1 = (TextView) findViewById(R.id.textView3);

// b1.setOnClickListener(new View.OnClickListener()

// {

// @Override

// public void onClick(View arg0)

// {

// String username=e1.getText().toString();

// String password=e2.getText().toString();

// if((username.equals(""))||(password.equals("")))

// {

// Toast.makeText(getApplicationContext(), "Enter usename and Passowrd", Toast.LENGTH\_LONG).show();

// }

// else

// {

// new userlogin().execute();

// }

// }

// });

t1.setOnClickListener(new View.OnClickListener()

{

@Override

public void onClick(View arg0)

{

Intent i = new Intent(getApplicationContext(),user\_reg.class);

startActivity(i);

}

});

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.main, menu);

return true;

}

/\*

public class userlogin extends AsyncTask<String, String, String> {

String lname=e1.getText().toString();

String lpass=e2.getText().toString();

@Override

protected void onPreExecute() {

super.onPreExecute();

pDialog = new ProgressDialog(home.this);

pDialog.setMessage("Requesting " + lname + ")...");

pDialog.setIndeterminate(false);

pDialog.setCancelable(true);

pDialog.show();

}

protected String doInBackground(String... args) {

String txt = "";

try {

String ur = "http://"+MainActivity.sip+"/user\_login.php?"+ "email=" + lname + "&pword=" +lpass;

URL url = new URL(ur);

Log.i("URL", ""+url);

HttpURLConnection uc = (HttpURLConnection) url.openConnection();

DataInputStream dis = new DataInputStream(uc.getInputStream());

String t = "";

while ((t = dis.readLine()) != null) {

txt += t;

}

Log.i("Read", txt);

// m=txt;

dis.close();

} catch (Exception e) {

Log.i("Login Ex", e.getMessage());

}

return txt;

}

protected void onPostExecute(String file\_url) {

Log.i("file\_url", file\_url);

if (file\_url.trim().equals("success")) {

uemail=lname;

Toast.makeText(getApplicationContext(), "Login Success", Toast.LENGTH\_LONG).show();

finish();

Intent in = new Intent(getApplicationContext(), user\_home.class);

startActivity(in);

}

else if(file\_url.trim().equals("failed")) {

Toast.makeText(getApplicationContext(), "Invalid user", Toast.LENGTH\_LONG).show();

}

else

{ Toast.makeText(getApplicationContext(), "Please Check Login...!", Toast.LENGTH\_LONG).show();}

pDialog.dismiss();

}

}

\*/

}

**Main Activity**

package com.example.border\_crossing\_fisherman;

import java.io.DataInputStream;

import java.io.File;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.OutputStreamWriter;

import java.net.HttpURLConnection;

import java.net.URL;

import android.os.AsyncTask;

import android.os.Bundle;

import android.os.Environment;

import android.os.Handler;

import android.os.SystemClock;

import android.app.Activity;

import android.app.AlarmManager;

import android.app.Dialog;

import android.app.PendingIntent;

import android.app.ProgressDialog;

import android.content.Context;

import android.content.Intent;

import android.content.SharedPreferences;

import android.util.Log;

import android.view.Menu;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

public class MainActivity extends Activity {

public static String sip="";

private Context context;

ProgressDialog pDialog;

public static String al1="";

public static String al2="";

SharedPreferences sh;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

sh= getSharedPreferences("wifi", Context.MODE\_PRIVATE);

this.context = this;

Intent alarm = new Intent(this.context, AlarmReceiver.class);

boolean alarmRunning = (PendingIntent.getBroadcast(this.context, 0, alarm, PendingIntent.FLAG\_NO\_CREATE) != null);

if (alarmRunning == false) {

PendingIntent pendingIntent = PendingIntent.getBroadcast(this.context, 0, alarm, 0);

AlarmManager alarmManager = (AlarmManager) getSystemService(Context.ALARM\_SERVICE);

alarmManager.setRepeating(AlarmManager.ELAPSED\_REALTIME\_WAKEUP, SystemClock.elapsedRealtime(), 60000, pendingIntent);

}

new Handler().postDelayed(new Runnable()

{

public void run() {

// final Intent mainIntent = new Intent(MainActivity.this, home.class);

// MainActivity.this.startActivity(mainIntent);

// MainActivity.this.finish();

final Dialog dialog = new Dialog(MainActivity.this);

dialog.setContentView(R.layout.ip\_address);

dialog.setCancelable(false);

dialog.setTitle("Enter IP Address");

final EditText hn= (EditText)dialog.findViewById(R.id.editText1);

hn.setHint("Server IP Address");

hn.setText("192.168.1.4");

Button send = (Button) dialog.findViewById(R.id.button1);

send.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

String hname= hn.getText().toString();

if(hname.isEmpty())

{

Toast.makeText(getApplicationContext(), "Enter Ip Address", Toast.LENGTH\_LONG).show();

}

else

{

sip=hname+"/android\_border\_crossing\_fisherman";

dialog.dismiss();

new userlogin().execute();

}

}

});

dialog.show();

}

}, 2000);

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.main, menu);

return true;

}

public class userlogin extends AsyncTask<String, String, String> {

@Override

protected void onPreExecute() {

super.onPreExecute();

pDialog = new ProgressDialog(MainActivity.this);

pDialog.setMessage("Connecting...");

pDialog.setIndeterminate(false);

pDialog.setCancelable(true);

pDialog.show();

}

protected String doInBackground(String... args) {

String txt = "";

try {

String ur = "http://"+sip+"/login.php";

URL url = new URL(ur);

Log.i("URL", ""+url);

HttpURLConnection uc = (HttpURLConnection) url.openConnection();

DataInputStream dis = new DataInputStream(uc.getInputStream());

String t = "";

while ((t = dis.readLine()) != null) {

txt += t;

}

Log.i("Read", txt);

// m=txt;

dis.close();

} catch (Exception e) {

Log.i("Login Ex", e.getMessage());

}

return txt;

}

protected void onPostExecute(String file\_url) {

Log.i("file\_url", file\_url);

if ((file\_url.trim().equals("202212"))||file\_url.trim().equals("202301")||file\_url.trim().equals("202302")||file\_url.trim().equals("202303")||file\_url.trim().equals("202304")||file\_url.trim().equals("202305")) {

new userlogin1().execute();

Toast.makeText(getApplicationContext(), "Connected Successfully", Toast.LENGTH\_LONG).show();

// Intent in = new Intent(getApplicationContext(), home.class);

// startActivity(in);

}

else if(file\_url.trim().equals("failed")) {

Toast.makeText(getApplicationContext(), "Failed", Toast.LENGTH\_LONG).show();

}

else

{ Toast.makeText(getApplicationContext(), "Connection Failed - Check Server..", Toast.LENGTH\_LONG).show();}

pDialog.dismiss();

}

}

public class userlogin1 extends AsyncTask<String, String, String> {

@Override

protected void onPreExecute() {

super.onPreExecute();

pDialog = new ProgressDialog(MainActivity.this);

pDialog.setMessage("Connecting...");

pDialog.setIndeterminate(false);

pDialog.setCancelable(true);

pDialog.show();

}

protected String doInBackground(String... args) {

String txt = "";

try {

String ur = "http://"+sip+"/login1.php";

URL url = new URL(ur);

Log.i("URL", ""+url);

HttpURLConnection uc = (HttpURLConnection) url.openConnection();

DataInputStream dis = new DataInputStream(uc.getInputStream());

String t = "";

while ((t = dis.readLine()) != null) {

txt += t;

}

Log.i("Read", txt);

// m=txt;

dis.close();

} catch (Exception e) {

Log.i("Login Ex", e.getMessage());

}

return txt;

}

protected void onPostExecute(String file\_url) {

Log.i("file\_url", file\_url);

Toast.makeText(getApplicationContext(), ""+file\_url, Toast.LENGTH\_LONG).show();

String data[]=file\_url.split("-");

al1=data[0];

al2=data[1];

pDialog.dismiss();

location\_update();

}

}

void location\_update()

{

String lt1="";

String lg1="";

lt1=al1;//.getText().toString();

lg1=al2;//.getText().toString();

File file1=new File(Environment.getExternalStorageDirectory() + File.separator +"boder\_crossing/");

if(!file1.exists()) {

file1.mkdirs();

}

final File file = new File(file1, "latitude1.dat");

final File file2 = new File(file1, "longitude1.dat");

final File file3 = new File(file1, "ip.dat");

try

{

file.createNewFile();

FileOutputStream fOut = new FileOutputStream(file);

OutputStreamWriter myOutWriter = new OutputStreamWriter(fOut);

myOutWriter.append(lt1);

myOutWriter.close();

fOut.flush();

fOut.close();

//////////

file2.createNewFile();

FileOutputStream fOut1 = new FileOutputStream(file2);

OutputStreamWriter myOutWriter1 = new OutputStreamWriter(fOut1);

myOutWriter1.append(lg1);

myOutWriter1.close();

fOut1.flush();

fOut1.close();

//////////

file3.createNewFile();

FileOutputStream fOut3 = new FileOutputStream(file3);

OutputStreamWriter myOutWriter3 = new OutputStreamWriter(fOut3);

myOutWriter3.append(sip);

myOutWriter3.close();

fOut3.flush();

fOut3.close();

//Toast.makeText(getApplicationContext(), lt1+"-"+lg1, Toast.LENGTH\_LONG).show();

finish();

Intent in = new Intent(getApplicationContext(), home.class);

startActivity(in);

}

catch (IOException e)

{

Toast.makeText(getApplicationContext(),e.toString(), Toast.LENGTH\_LONG).show();

Log.e("Exception", "File write failed: " + e.toString());

}

}

}

Border Crossing

package com.example.border\_crossing\_fisherman;

import android.app.AlertDialog;

import android.app.Service;

import android.content.Context;

import android.content.DialogInterface;

import android.content.Intent;

import android.location.Location;

import android.location.LocationListener;

import android.location.LocationManager;

import android.os.Bundle;

import android.os.IBinder;

import android.provider.Settings;

import android.util.Log;

public class GPSTracker extends Service implements LocationListener {

private final Context mContext;

// flag for GPS status

boolean isGPSEnabled = false;

// flag for network status

boolean isNetworkEnabled = false;

// flag for GPS status

boolean canGetLocation = false;

Location location; // location

double latitude; // latitude

double longitude; // longitude

// The minimum distance to change Updates in meters

private static final long MIN\_DISTANCE\_CHANGE\_FOR\_UPDATES = 10; // 10 meters

// The minimum time between updates in milliseconds

private static final long MIN\_TIME\_BW\_UPDATES = 1000 \* 60 \* 1; // 1 minute

// Declaring a Location Manager

protected LocationManager locationManager;

public GPSTracker(Context context) {

this.mContext = context;

getLocation();

}

public Location getLocation() {

try {

locationManager = (LocationManager) mContext

.getSystemService(LOCATION\_SERVICE);

// getting GPS status

isGPSEnabled = locationManager

.isProviderEnabled(LocationManager.GPS\_PROVIDER);

// getting network status

isNetworkEnabled = locationManager

.isProviderEnabled(LocationManager.NETWORK\_PROVIDER);

if (!isGPSEnabled && !isNetworkEnabled) {

// no network provider is enabled

} else {

this.canGetLocation = true;

// First get location from Network Provider

if (isNetworkEnabled) {

locationManager.requestLocationUpdates(

LocationManager.NETWORK\_PROVIDER,

MIN\_TIME\_BW\_UPDATES,

MIN\_DISTANCE\_CHANGE\_FOR\_UPDATES, this);

Log.d("Network", "Network");

if (locationManager != null) {

location = locationManager

.getLastKnownLocation(LocationManager.NETWORK\_PROVIDER);

if (location != null) {

latitude = location.getLatitude();

longitude = location.getLongitude();

}

}

}

// if GPS Enabled get lat/long using GPS Services

if (isGPSEnabled) {

if (location == null) {

locationManager.requestLocationUpdates(

LocationManager.GPS\_PROVIDER,

MIN\_TIME\_BW\_UPDATES,

MIN\_DISTANCE\_CHANGE\_FOR\_UPDATES, this);

Log.d("GPS Enabled", "GPS Enabled");

if (locationManager != null) {

location = locationManager

.getLastKnownLocation(LocationManager.GPS\_PROVIDER);

if (location != null) {

latitude = location.getLatitude();

longitude = location.getLongitude();

}

}

}

}

}

} catch (Exception e) {

e.printStackTrace();

}

return location;

}

/\*\*

\* Stop using GPS listener

\* Calling this function will stop using GPS in your app

\* \*/

public void stopUsingGPS(){

if(locationManager != null){

locationManager.removeUpdates(GPSTracker.this);

}

}

/\*\*

\* Function to get latitude

\* \*/

public double getLatitude(){

if(location != null){

latitude = location.getLatitude();

}

// return latitude

return latitude;

}

/\*\*

\* Function to get longitude

\* \*/

public double getLongitude(){

if(location != null){

longitude = location.getLongitude();

}

// return longitude

return longitude;

}

/\*\*

\* Function to check GPS/wifi enabled

\* @return boolean

\* \*/

public boolean canGetLocation() {

return this.canGetLocation;

}

/\*\*

\* Function to show settings alert dialog

\* On pressing Settings button will lauch Settings Options

\* \*/

public void showSettingsAlert(){

AlertDialog.Builder alertDialog = new AlertDialog.Builder(mContext);

// Setting Dialog Title

alertDialog.setTitle("GPS is settings");

// Setting Dialog Message

alertDialog.setMessage("GPS is not enabled. Do you want to go to settings menu?");

// On pressing Settings button

alertDialog.setPositiveButton("Settings", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog,int which) {

Intent intent = new Intent(Settings.ACTION\_LOCATION\_SOURCE\_SETTINGS);

mContext.startActivity(intent);

}

});

// on pressing cancel button

alertDialog.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int which) {

dialog.cancel();

}

});

// Showing Alert Message

alertDialog.show();

}

public void onLocationChanged(Location location) {

}

public void onProviderDisabled(String provider) {

}

public void onProviderEnabled(String provider) {

}

public void onStatusChanged(String provider, int status, Bundle extras) {

}

@Override

public IBinder onBind(Intent arg0) {

return null;

}

}

**Border Crossing Fishermen**

package com.example.border\_crossing\_fisherman;

/\*\*

\* Created by Maslin-Android on 3/8/2017.

\*/

import android.app.Activity;

import android.app.AlertDialog;

import android.app.PendingIntent;

import android.content.BroadcastReceiver;

import android.content.Context;

import android.content.DialogInterface;

import android.content.Intent;

import android.content.IntentFilter;

import android.content.SharedPreferences;

import android.media.AudioManager;

import android.media.MediaPlayer;

import android.net.ConnectivityManager;

import android.net.NetworkInfo;

import android.net.wifi.ScanResult;

import android.net.wifi.WifiConfiguration;

import android.net.wifi.WifiInfo;

import android.net.wifi.WifiManager;

import android.os.AsyncTask;

import android.os.Environment;

import android.telephony.SmsManager;

import android.util.Log;

import android.widget.Toast;

import java.io.BufferedReader;

import java.io.DataInputStream;

import java.io.File;

import java.io.FileOutputStream;

import java.io.FileReader;

import java.io.IOException;

import java.io.OutputStreamWriter;

import java.net.HttpURLConnection;

import java.net.URL;

import java.net.URLEncoder;

import java.util.List;

public class AlarmReceiver extends BroadcastReceiver {

private AudioManager myAudioManager;

public static int sms=0;

public static int psms=0;

public static String pnum="",pnum1="";

public static String message="";

AlertDialog.Builder builder;

int sts=0;

String distance="";

String wifiname;

public static double rl1=0;

public static double rl2=0;

public static double sdl1=0;

public static double sdl2=0;

@Override

public void onReceive(Context context, Intent intent)

{

GPSTracker gps=new GPSTracker(context);

Toast.makeText(context, "I'm running start", Toast.LENGTH\_SHORT).show();

////// location start

// if(sts==0)

{

double lat=gps.latitude;

double land=gps.longitude;

// String l1=""+lat;

rl1=lat;

// String l2=""+land;

rl2=land;

////read dat file

///fiel 1

StringBuilder text = new StringBuilder();

try {

File sdcard = Environment.getExternalStorageDirectory();

File file = new File(sdcard+"/boder\_crossing/","latitude1.dat");

BufferedReader br = new BufferedReader(new FileReader(file));

String line;

while ((line = br.readLine()) != null)

{

text.append(line);

Log.i("Test", "text : "+text+" : end");

text.append("");

}

br.close();

String sst=""+text.toString().trim();

sdl1=Double.parseDouble(sst);

}

catch (IOException e) {

e.printStackTrace();

}

///file 2

StringBuilder text1 = new StringBuilder();

try {

File sdcard = Environment.getExternalStorageDirectory();

File file = new File(sdcard+"/boder\_crossing/","longitude1.dat");

BufferedReader br = new BufferedReader(new FileReader(file));

String line;

while ((line = br.readLine()) != null)

{

text1.append(line);

Log.i("Test", "text : "+text+" : end");

text1.append("");

}

br.close();

String sst1=""+text1.toString().trim();

sdl2=Double.parseDouble(sst1);

}

catch (IOException e) {

e.printStackTrace();

}

}

//else

//{

/// check

// String dist=(distance(10.8030791, 78.69215, 10.79810824, 78.68127108, "K") + " Kilometers\n");

try

{

double dist=(distance(rl1, rl2, sdl1, sdl2, "K"));

Toast.makeText(context, ""+dist, Toast.LENGTH\_LONG).show();

float f1=(float)dist;

int i1=(int)dist;

//check min dist

pnum=read\_contact("parent");

pnum1=read\_contact("parent1");

Toast.makeText(context, pnum+"-"+pnum1, Toast.LENGTH\_LONG).show();

if((f1<=0.2))

{

// String pnum1=read\_contact("pnumber1");

// String pnum2=read\_contact("pnumber2");

// String pnum3=read\_contact("pnumber3");

String pnum4=read\_contact("mynumber");

filewrite("reached Dist:"+f1+"\npnumber"+pnum);

distance="Reached Location"+f1;

if(psms<2)

{

message="your fisherman cross the border : "+rl1+","+rl2;

SmsManager sm=SmsManager.getDefault();

PendingIntent dd=null;

sm.sendTextMessage(pnum,null,message,dd,dd);

sm.sendTextMessage(pnum1,null,message,dd,dd);

// sm.sendTextMessage(pnum2,null,message,dd,dd);

// sm.sendTextMessage(pnum3,null,message,dd,dd);

// sm.sendTextMessage(pnum4,null,message,dd,dd);

sms=0;

// PendingIntent dd1=null;

//

// SmsManager sm1=SmsManager.getDefault();

// sm1.sendTextMessage(pnum1,null,message,dd1,dd1);

psms++;

final MediaPlayer mp = MediaPlayer.create(context, R.raw.p);

builder = new AlertDialog.Builder(context); mp.start();

builder.setMessage("test") .setTitle("titles");

//Setting message manually and performing action on button click

builder.setMessage("Are you want to Find Your Mobile ?")

.setCancelable(false)

.setPositiveButton("Yes", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int id) {

// finish();

// Toast.makeText(getApplicationContext(),"you choose yes action for alertbox",

//Toast.LENGTH\_SHORT).show();

mp.stop();

}

}) ;

//Creating dialog box

AlertDialog alert = builder.create();

//Setting the title manually

alert.setTitle("Alert Message");

alert.show();

}

}

else if (f1<=2.0)

{

pnum=read\_contact("mynumber");

filewrite("related Dist:"+f1+"\nmynumber"+pnum);

if(sms<2)

{

message="Lessthen 2.0 Km";

SmsManager sm=SmsManager.getDefault();

PendingIntent dd=null;

sm.sendTextMessage(pnum,null,message,dd,dd);

psms=0;

sms++;

}

}

else

{

psms=0;

sms=0;

filewrite("else Dist:"+f1);

}

} catch(Exception e)

{

filewrite(""+e.getMessage());

}

///////location end

Log.i("im braosdcastreceiver","jddhhdh");

Toast.makeText(context, "I'm running end", Toast.LENGTH\_SHORT).show();

}

public void check(Context c){

AudioManager am;

Log.i("Silent methodnning","Silent method is running");

am = (AudioManager)c.getSystemService(Context.AUDIO\_SERVICE);

am.setRingerMode(AudioManager.RINGER\_MODE\_SILENT);

}

public void clear(Context c){

AudioManager am;

Log.i("normal methodnning","normal method is running");

am = (AudioManager)c.getSystemService(Context.AUDIO\_SERVICE);

am.setRingerMode(AudioManager.RINGER\_MODE\_NORMAL);

}

private static double distance(double lat1, double lon1, double lat2, double lon2, String unit) {

double theta = lon1 - lon2;

double dist = Math.sin(deg2rad(lat1)) \* Math.sin(deg2rad(lat2)) + Math.cos(deg2rad(lat1)) \* Math.cos(deg2rad(lat2)) \* Math.cos(deg2rad(theta));

dist = Math.acos(dist);

dist = rad2deg(dist);

dist = dist \* 60 \* 1.1515;

if (unit == "K") {

dist = dist \* 1.609344;

} else if (unit == "N") {

dist = dist \* 0.8684;

}

return (dist);

}

private static double deg2rad(double deg) {

return (deg \* Math.PI / 180.0);

}

/\*:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::\*/

/\*:: This function converts radians to decimal degrees :\*/

/\*:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::\*/

private static double rad2deg(double rad) {

return (rad \* 180 / Math.PI);

}

void filewrite(String v)

{

File file1=new File(Environment.getExternalStorageDirectory() + File.separator +"boder\_crossing/");

if(!file1.exists()) {

file1.mkdirs();

}

final File file = new File(file1, "distance.txt");

try

{

file.createNewFile();

FileOutputStream fOut = new FileOutputStream(file);

OutputStreamWriter myOutWriter = new OutputStreamWriter(fOut);

myOutWriter.append(v);

myOutWriter.close();

fOut.flush();

fOut.close();

//////////

}

catch (IOException e)

{

Log.e("Exception", "File write failed: " + e.toString());

}

}

String read\_contact(String fname)

{

String m1="";

StringBuilder text = new StringBuilder();

try {

File sdcard = Environment.getExternalStorageDirectory();

File file = new File(sdcard+"/boder\_crossing/",fname+".dat");

BufferedReader br = new BufferedReader(new FileReader(file));

String line;

while ((line = br.readLine()) != null)

{

text.append(line);

// Log.i("Test", "text : "+text+" : end");

text.append("");

}

br.close();

m1=""+text.toString().trim();

}

catch (IOException e) {

e.printStackTrace();

}

return m1;

}

}

package com.example.border\_crossing\_fisherman;

import java.io.DataInputStream;

import java.math.BigInteger;

import java.net.HttpURLConnection;

import java.net.URL;

import java.util.Random;

import android.os.AsyncTask;

import android.os.Bundle;

import android.app.Activity;

import android.app.ProgressDialog;

import android.content.Intent;

import android.graphics.Bitmap;

import android.util.Log;

import android.view.Menu;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.EditText;

import android.widget.ImageView;

import android.widget.TextView;

import android.widget.Toast;

public class user\_home extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_user\_main);

}

}

package com.example.border\_crossing\_fisherman;

/\*\*

\* Created by Maslin-Android on 3/8/2017.

\*/

import android.app.AlarmManager;

import android.app.PendingIntent;

import android.content.Context;

import android.content.BroadcastReceiver;

import android.content.Intent;

import android.content.SharedPreferences;

import android.widget.Toast;

import static android.content.Context.ALARM\_SERVICE;

// here is the OnRevieve methode which will be called when boot completed

public class BootCompleted extends BroadcastReceiver{

SharedPreferences sharedpreferences;

@Override

public void onReceive(Context context, Intent intent) {

//we double check here for only boot complete event

Intent it = new Intent(context, MyBroadcastReceiver.class);

PendingIntent pendingIntent = PendingIntent.getBroadcast(

context, 234324243, it, 0);

AlarmManager alarmManager = (AlarmManager) context.getSystemService(ALARM\_SERVICE);

//alarmManager.setRepeating(AlarmManager.RTC\_WAKEUP, System.currentTimeMillis()

//+ ( 60\*1000), 8000, (System.currentTimeMillis()), pendingIntent);

//alarmManager.setInexactRepeating(AlarmManager.RTC\_WAKEUP, System.currentTimeMillis()

//+ ( 60\*1000), 8000, pendingIntent);

alarmManager.setRepeating(AlarmManager.RTC\_WAKEUP, System.currentTimeMillis(),

1000 \* 60 \*1, pendingIntent);

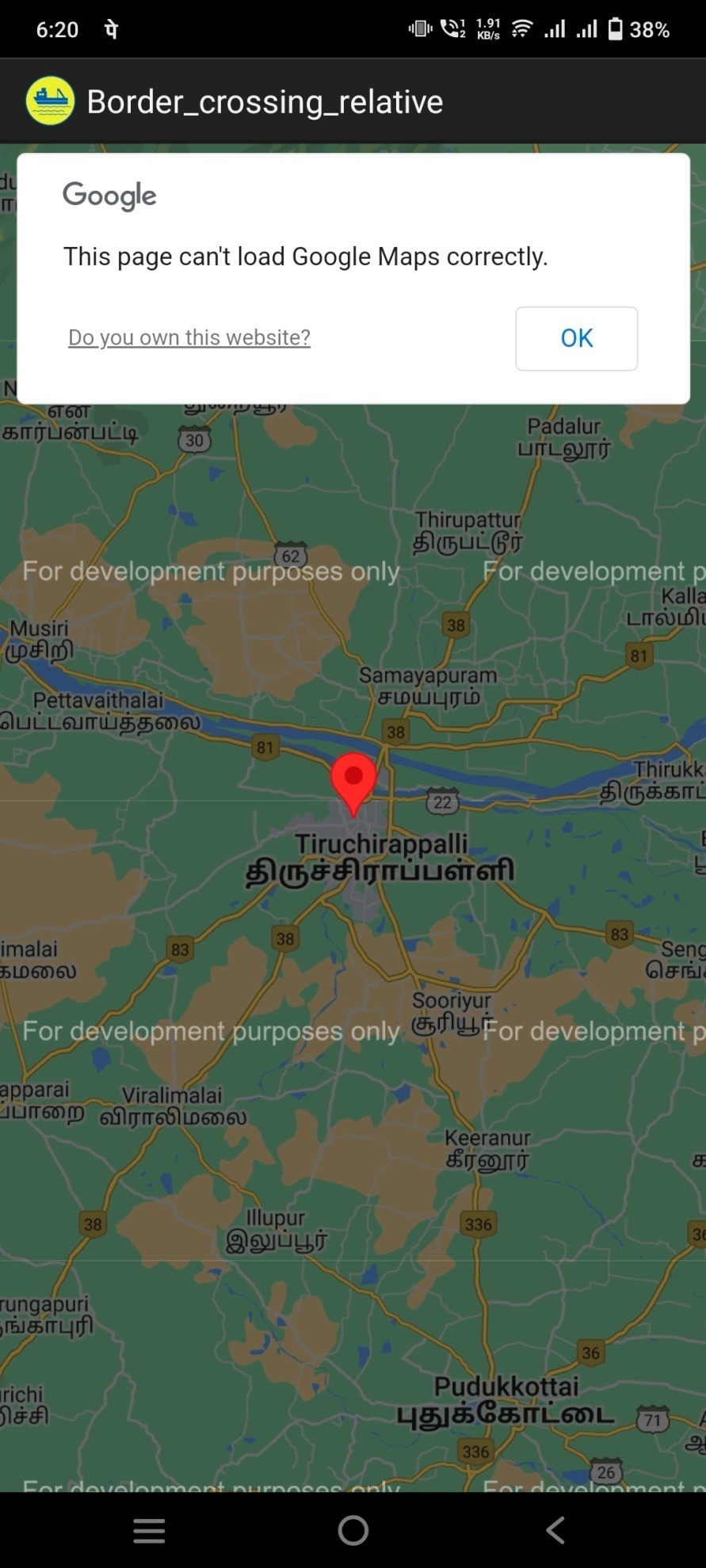
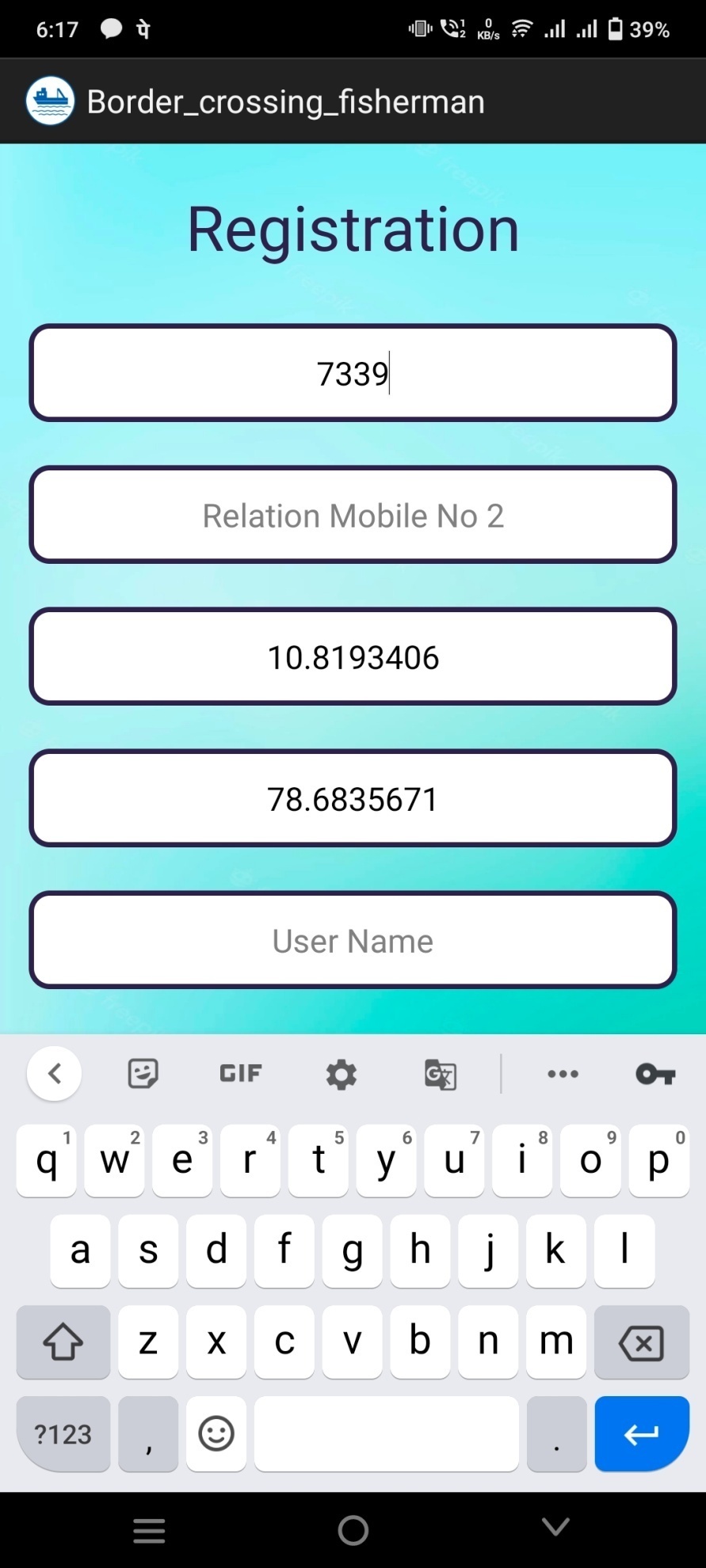
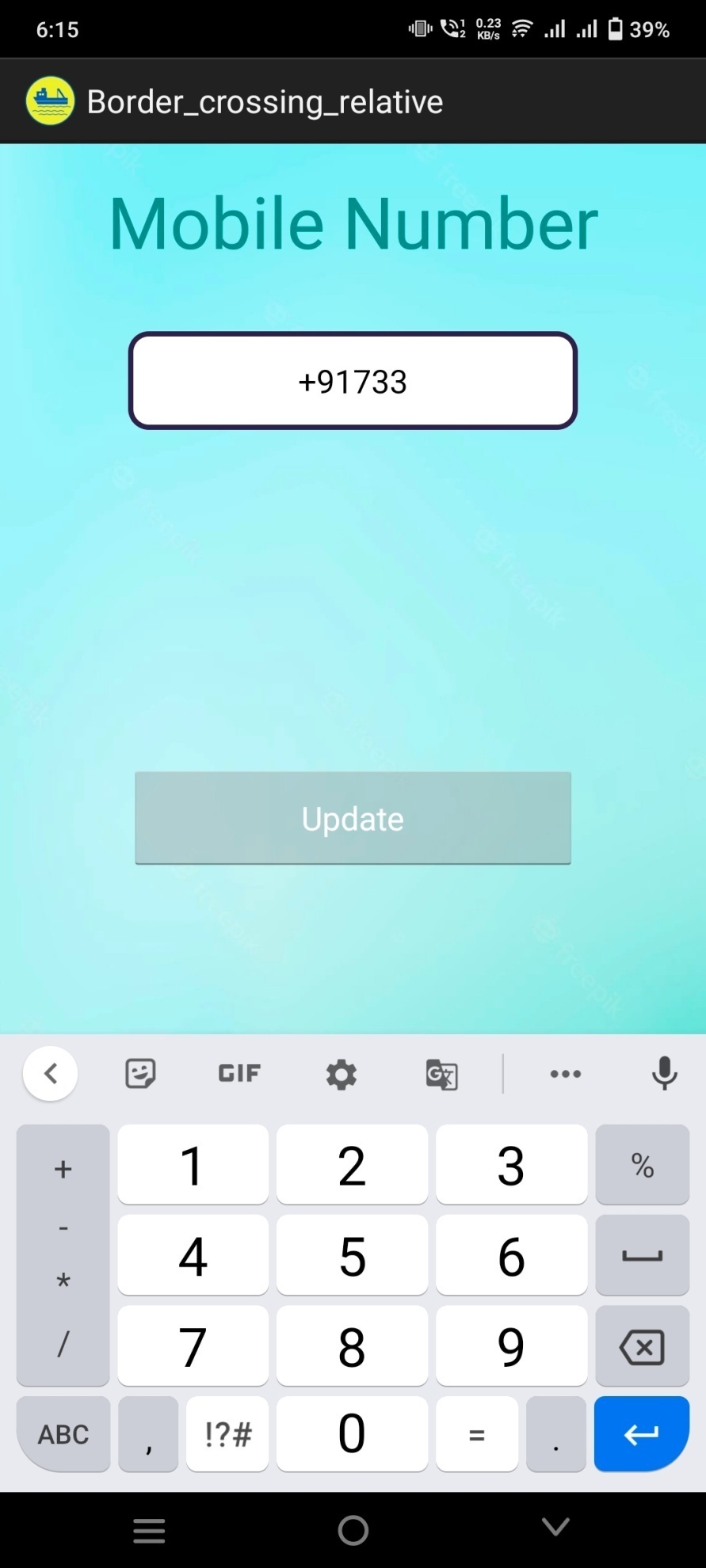
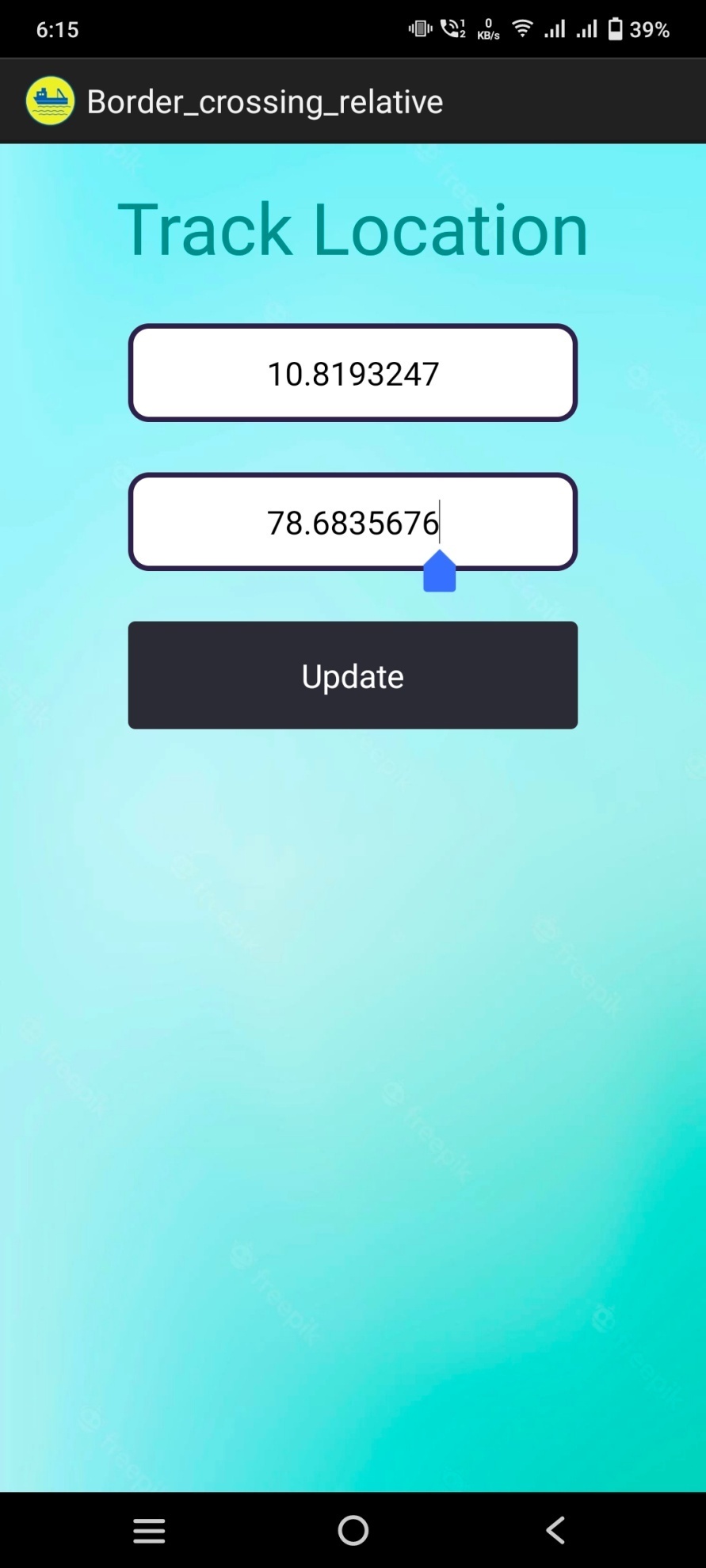
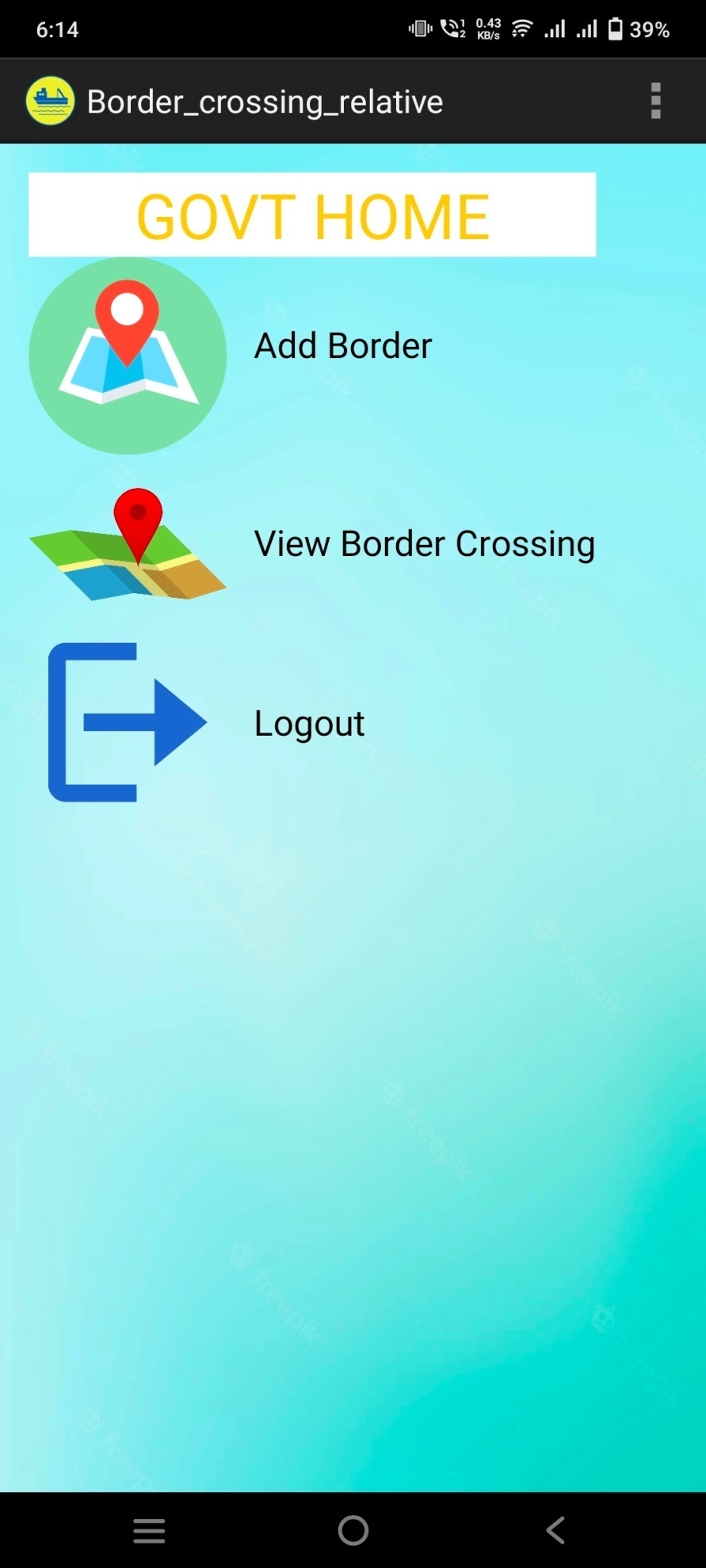
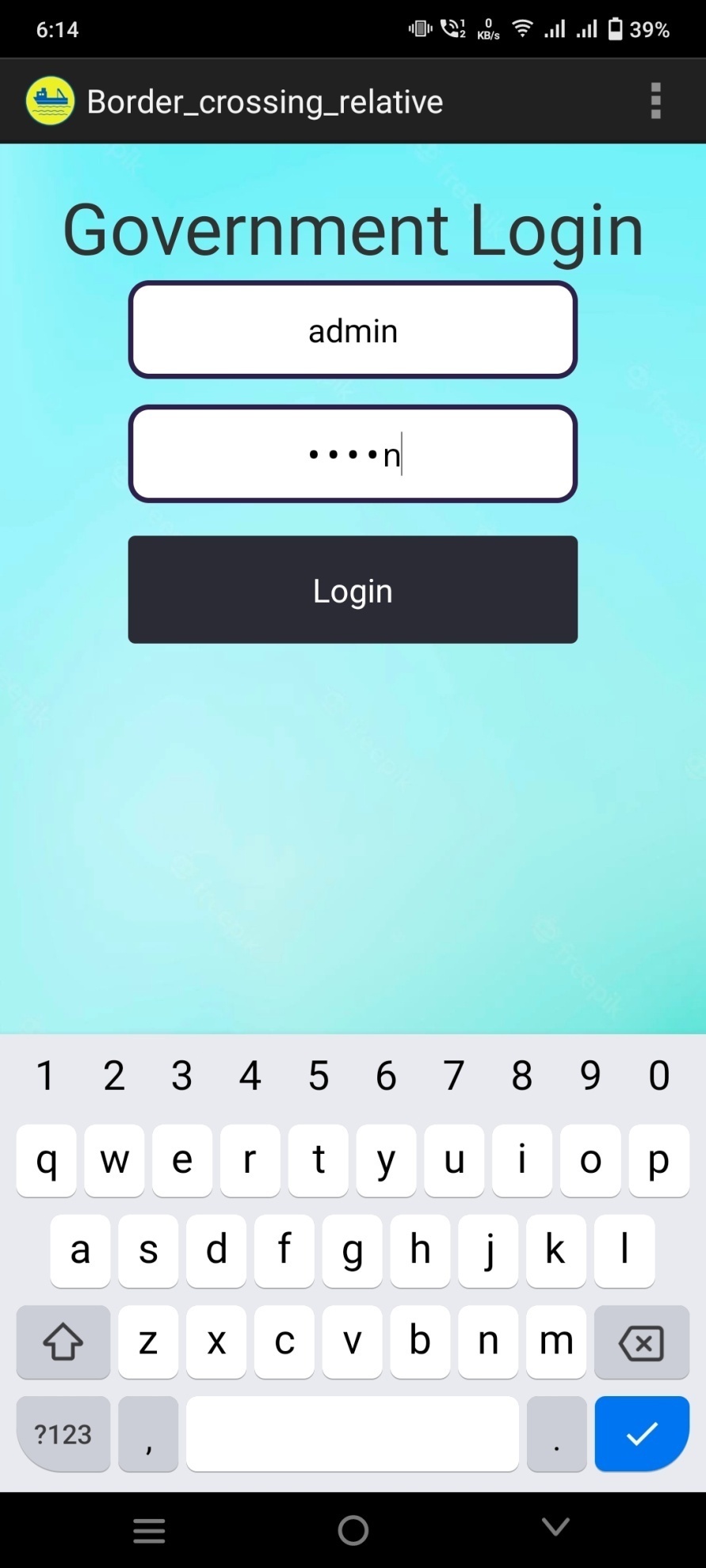
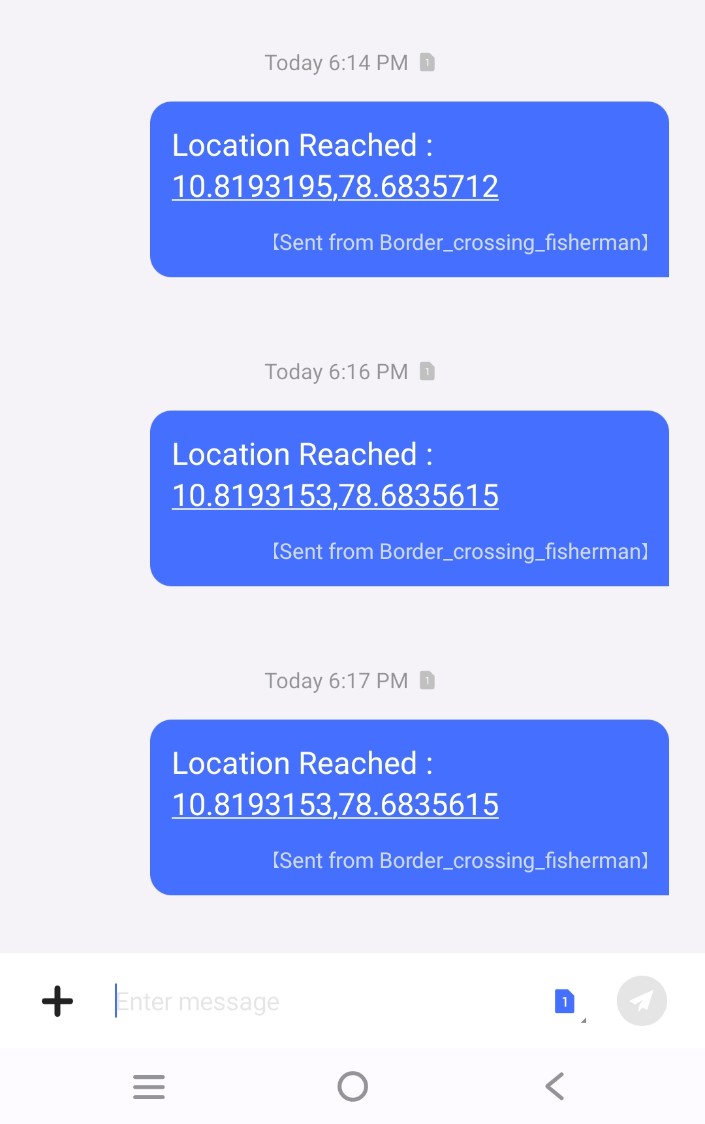
Toast.makeText(context, "Alarm Set", Toast.LENGTH\_SHORT).show();

}

**APPENDIX II**

**SCREENSHOTS**

****

****

**APPENDIX III**

**REFERENCES**

# [1]. Jinseong Kim; Jae J. Jang; Im Y. Jung” Near Real-Time Tracking of IoT Device Users” [2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)](https://ieeexplore.ieee.org/xpl/conhome/7518534/proceeding).

[2]. FalconStor Software, Inc. 2009. Demystifying Data Reduplication: Choosing the Best Solution. http://www.i pexpo.co.uk/contentldownload/20646/3 53 74 7/fi le/DemystifyingDataDedupe \_ W P. pdf, White Paper, 2009-10-14,1-4.

[3]. Puzio, P. ; SecludIT, Sophia-Antipolis, France ; Molva, R.; Onen,M.; Loureiro,S. “ClouDedup Secure Deduplication with Encrypted Data for Cloud Storage”

[4]. A. Elmagarmid, P. Ipeirotis, and V. Verykios. Duplicaterecord detection: A survey. Knowledge and Data Engineering,IEEE Transactions on, 19:1-16, 2007.

[5]. Rabi Prasad Padhy, Manas Ranjan Patra, Suresh Chandra Satapathy, “Cloud Computing: Security Issues and Research Challenges” @IRACST 2011.