# Tracking Phone System

**Shukatullah Anwari**
**1900904**
**Hussein Aldahesh**
**1801959**
**Boran Akkaya**
**1803430**

20 MAY 2021

—

Data Structures & Algorithms II

—

Merve ARITÜRK

## Introduction

Technology is advancing with a tremendous speed nowadays and every single member of technology community has at least one mobile phone and demands of mobile phone is increasing as year goes by due to its usefulness in our daily life as a day without using mobile phone will be felt like weeks if not a month but what if a day comes where you lose your mobile phone, or it has been stolen. Here comes our Track Phone system which ease our user to keep track of their phone in case of misfortune.


## Project Definition:

Tracking Phone System (TPS) helps the user to remote control and keep track of his/her smart phone by control and locate the live location, history, last location, and send as well as receive data. The system is design in such a way to help the user in case of lost phone or stolen to keep track of live location that shows the current location of the phone along with any movement in real-time. In case of disconnect, it will help the user to go through history to location the last whereabout of the phone before being disconnect and find where the user's phone has been along with the date and time. The user can download TPS as an application through many distribution platforms such as App Store, Play Store and many others, also it comes as original system which will be used in PCs too. The system is developed using JAVA. The project will be build using 2 data structure that are double-linked list and Tree. In the content of the project, the group will design a GUI to show all modules about TPS.

## TPS Summary

Tracking Phone System (TPS) is build using JAVA and it has five features that will help the user to keep better track of his/her phone. First of all, the application will have a sign in and sign up page which register the user to have his/her own private account and access to TPS. It will be the first page that will show on the screen and the user has to sign in in order to use the application, this will help you to keep the user data safe and no one else to misuse the application and change data and details. Once the user login process is complete, the application will take you to the main page of TPS where there be couple of option such as Registration, Tracking, Blocking, Warning/Message, and History.

In Registration the user can register his/her detail along with phone's detail. In this page, the user will be asked to fill the following, Phone IMEI number, Name, Phone number, Email, Gender, Address, as well as the system will generate a unique serial number each time a new phone registered. This page will help the user to save his/her phone's detail in order to use the TPS's other features.

In Tracking the user can track his/her phone's live location using the serial number or IMEI number. In this page the user has to enter either Serial Number or IMEI number and the system will locate the phone using google map to trace and keep track of the phone.

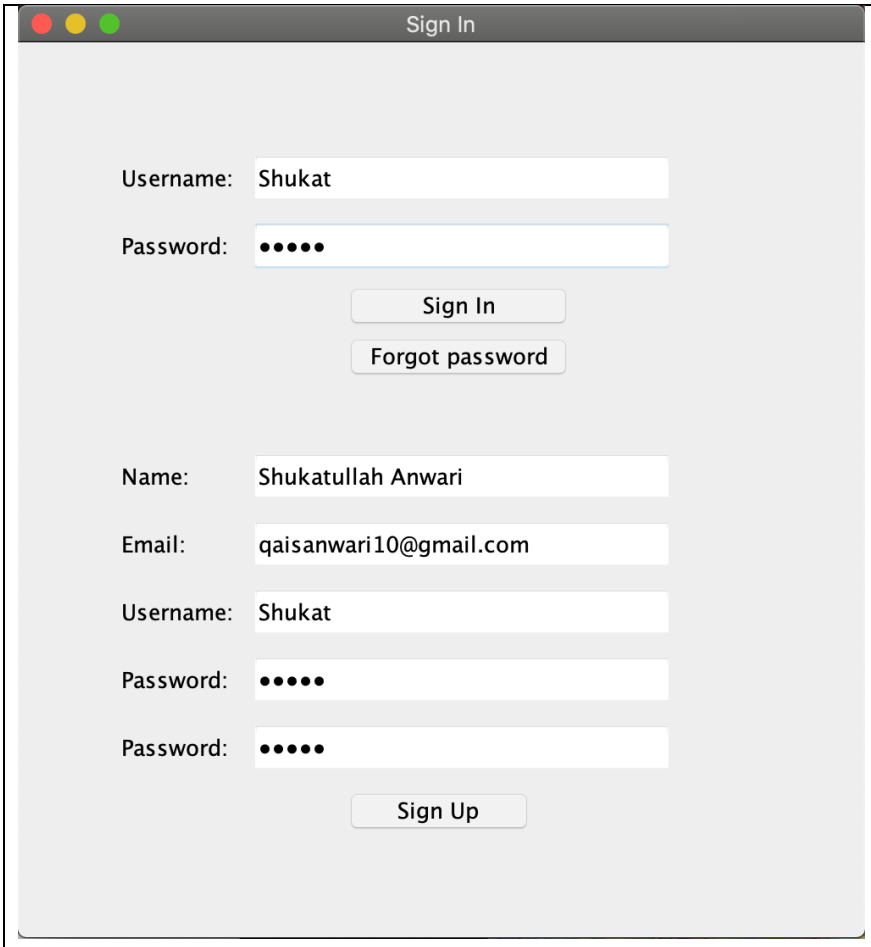
In Blocking the user will know the state of his/her phone whether its Block or Active and user can block or active his/her phone using the serial number.

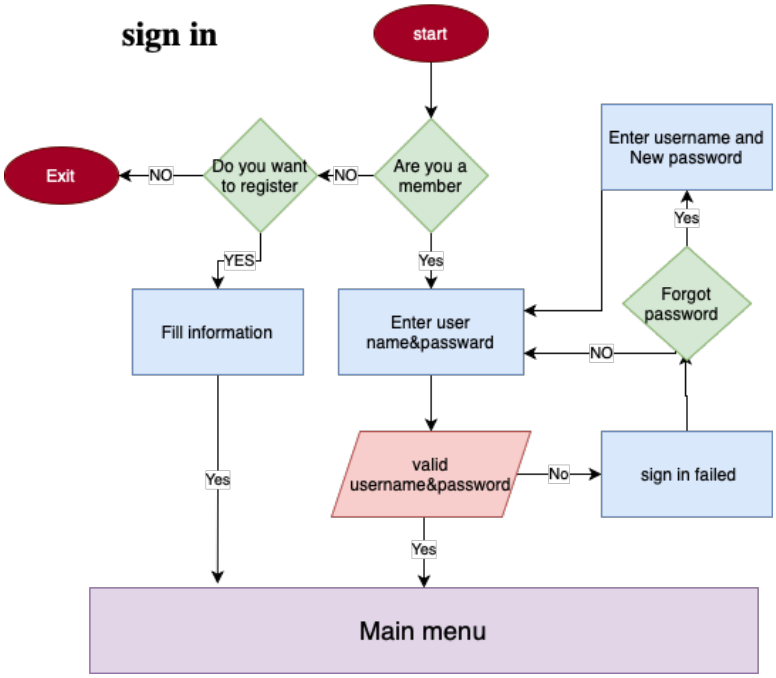
In Warning/Message the user can send message or a warning to his/her phone in case of lost phone or stolen. In this page the user has to enter either Serial Number or IMEI number to send message or a warning.

In History the user can keep track of the phone's history. In the history the user has to enter either Serial Number or IMEI number in order to track his/her phone's last location, date & time, phone number, and name.

# Project details, Flowcharts, GUI design and Source Code

## Sign In /Sign Up Page



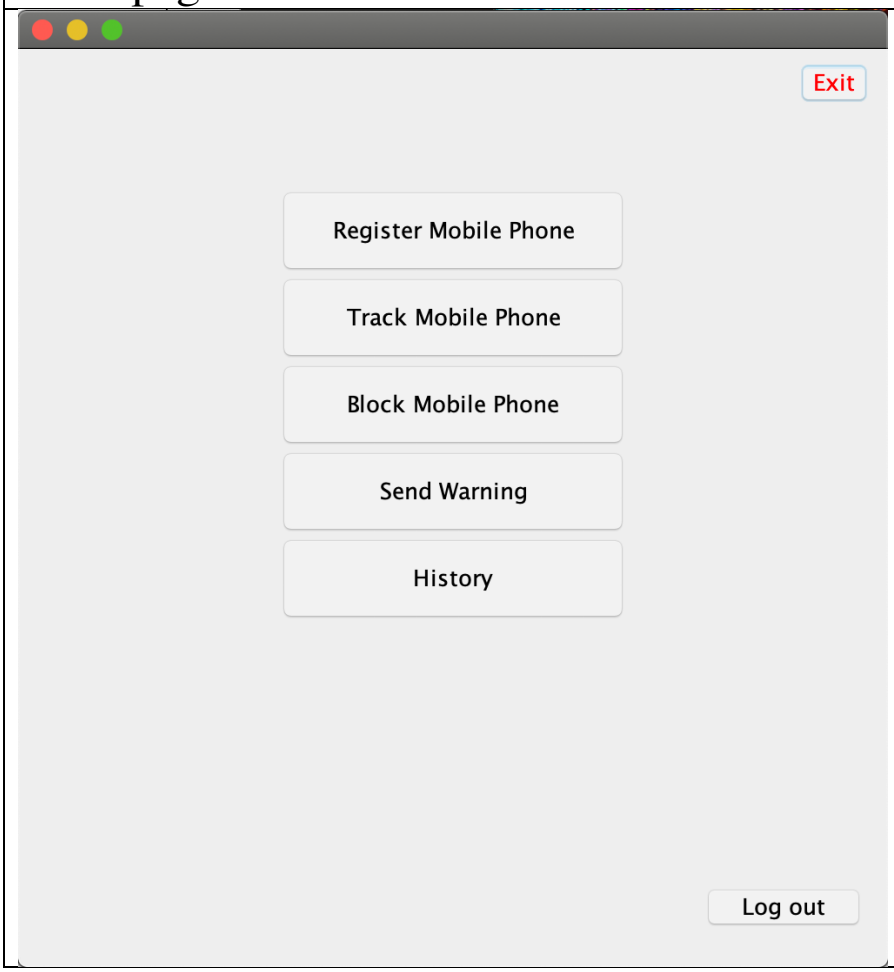

Once the application is open then the user will see the figure on the left. Here the user has to sign in or register as a new member so that he/she can access to the app and enter the main page of the application. The Sign In page has a total of 3 button as such as Sign in, Forgot Password & Sign up.

Sign in button will check the Input, Username & password, from the user and compare it in the database with the registered members. If the database matches, then it will process to main menu, if not then it will give a message saying, “Invalid user or password”.

Forgot password button will help the user who forgot his/her password to change the new password.

Sign up button will check the input information given from the user and help compare the password and register the new member to the database and save it so that the user can sign in the system.

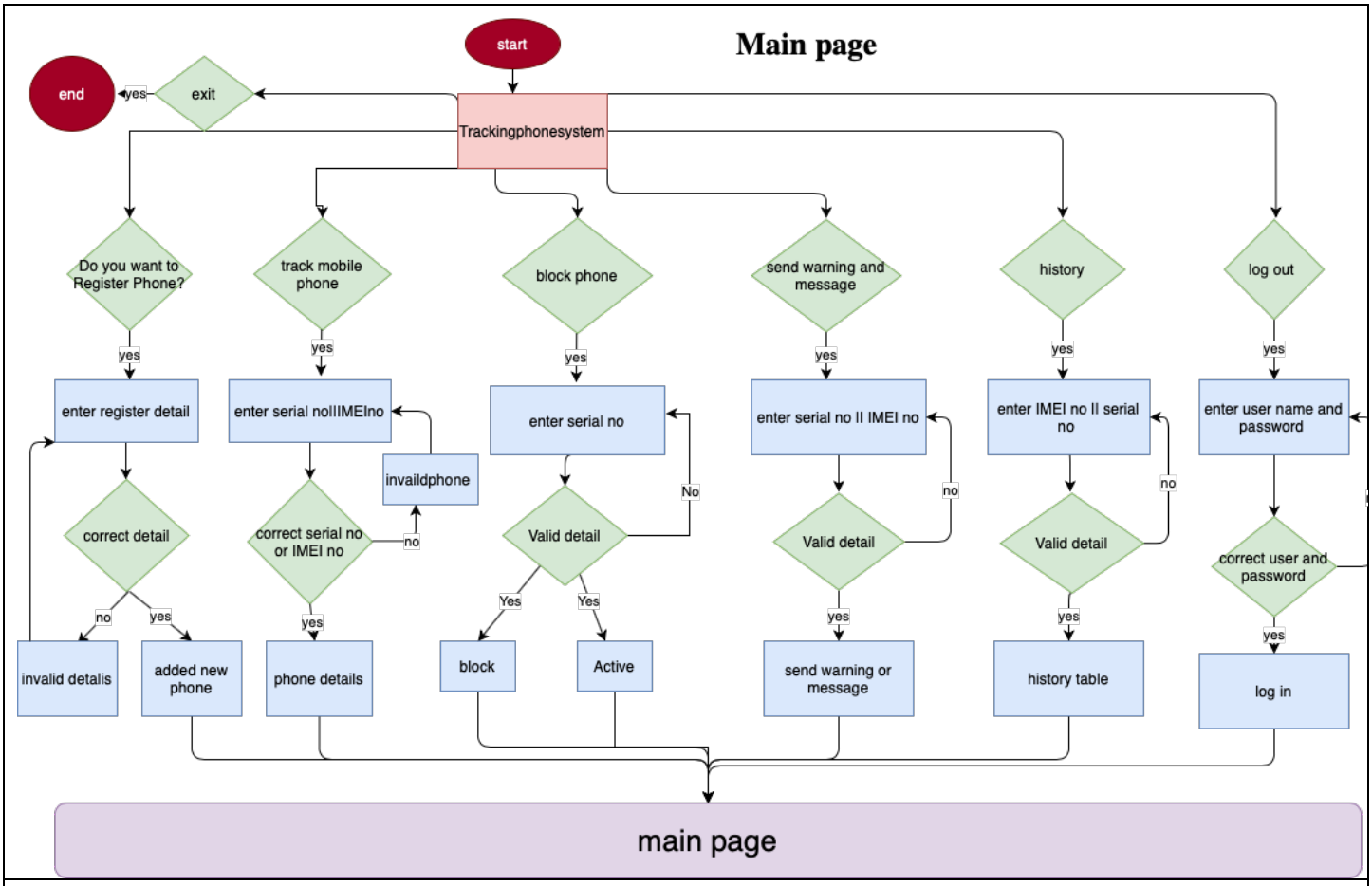
# Main page



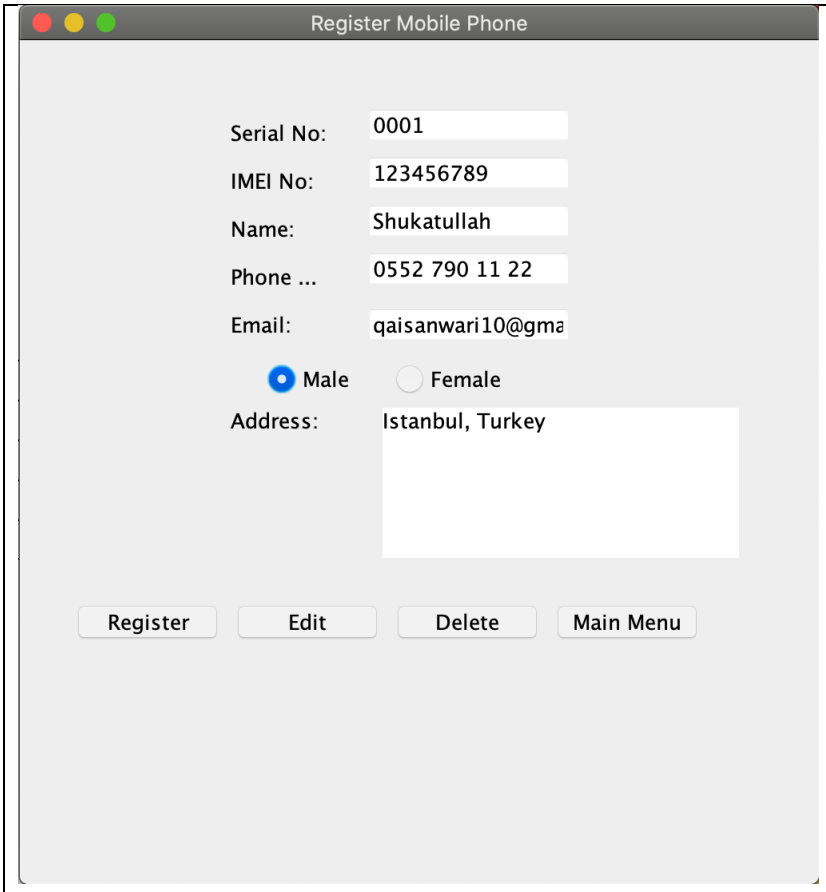

The Main page will contain of five option,Register Mobile Phone, Track Mobile Phone, Block Mobile Phone, Send Warning and History where the user will pick any of the following option mention on the right side in order to keep track of his/her mobile phone. It will also have an exit button and log out button.

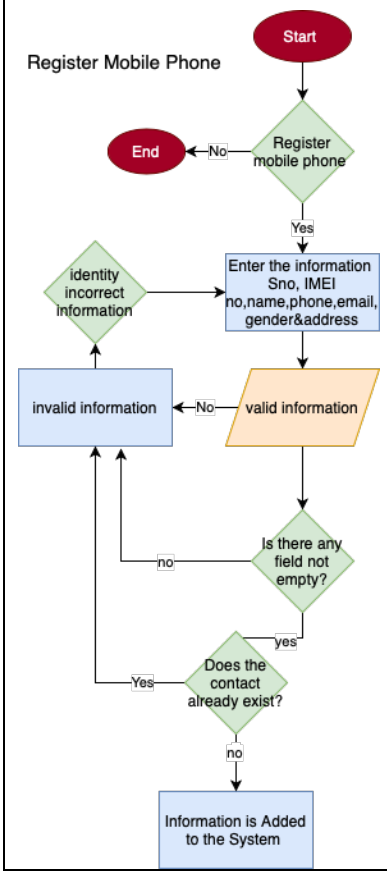
The Exit button will close the whole system.

The log out button will take the user back to the sign in page where the user has to sign in again in order to come back to main menu or user can register a new member.




Register Mobile Phone page

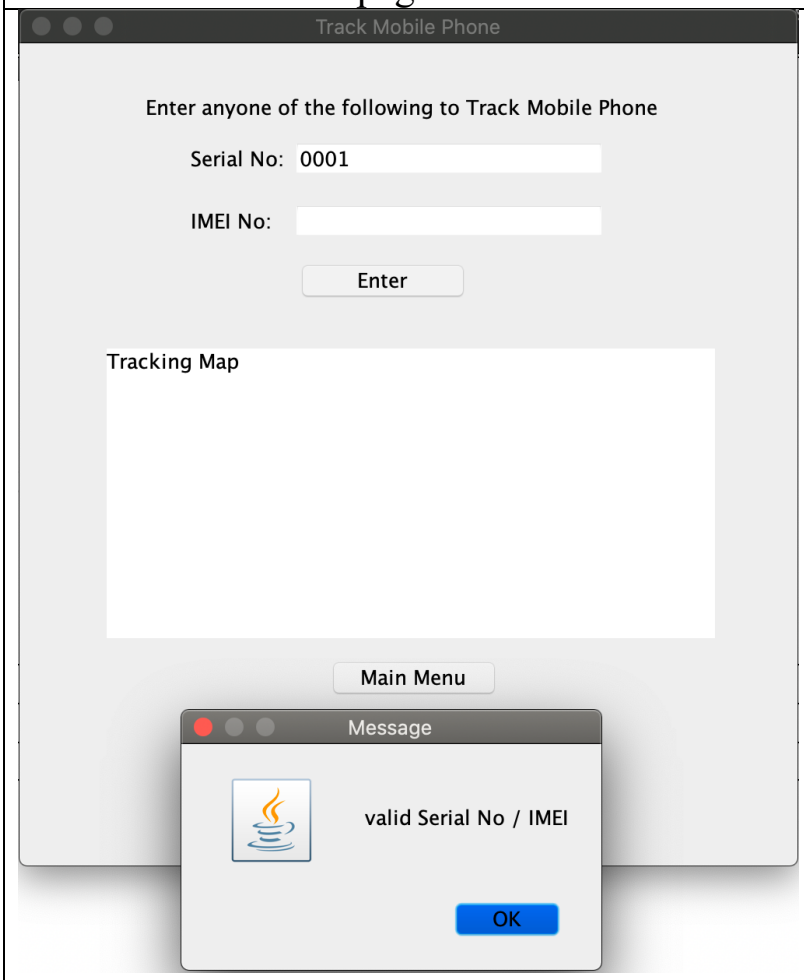


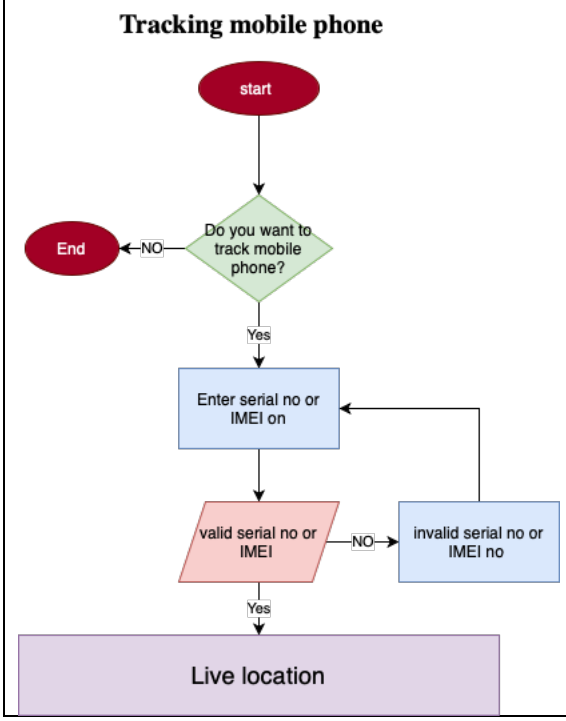




As shown in the figure to the side, once the Register Mobile Phone page is select then a new screen will show and it will ask you to add mobile phone's detail. The details includes 1.Serial no which will be generated by the system,, 2.IMEI no which will be input by the user which a unique identity of each phone, 3.Name which the user can input in order to know identity of the phone that has been registered, 4.Phone Number helps the user to keep tracking of the phone numbers, 5.Email which will be used to send or received information of the phone to the user, 6.gender, and 7.Address. once it is enter and the register button is clicked then the system will check if the detail is vaild or not and if it is already existed or not then It will add the contact and detail of the phone in the system. The Page has 3 more button which help the user to have to Edit existent information or the delete it and it has a main menu button which will return back to main page.

The register mobile phone page will contain the binary tree source code and the system will use the binary tree to register the new phone's detail in the system.

## Track Mobile Phone page

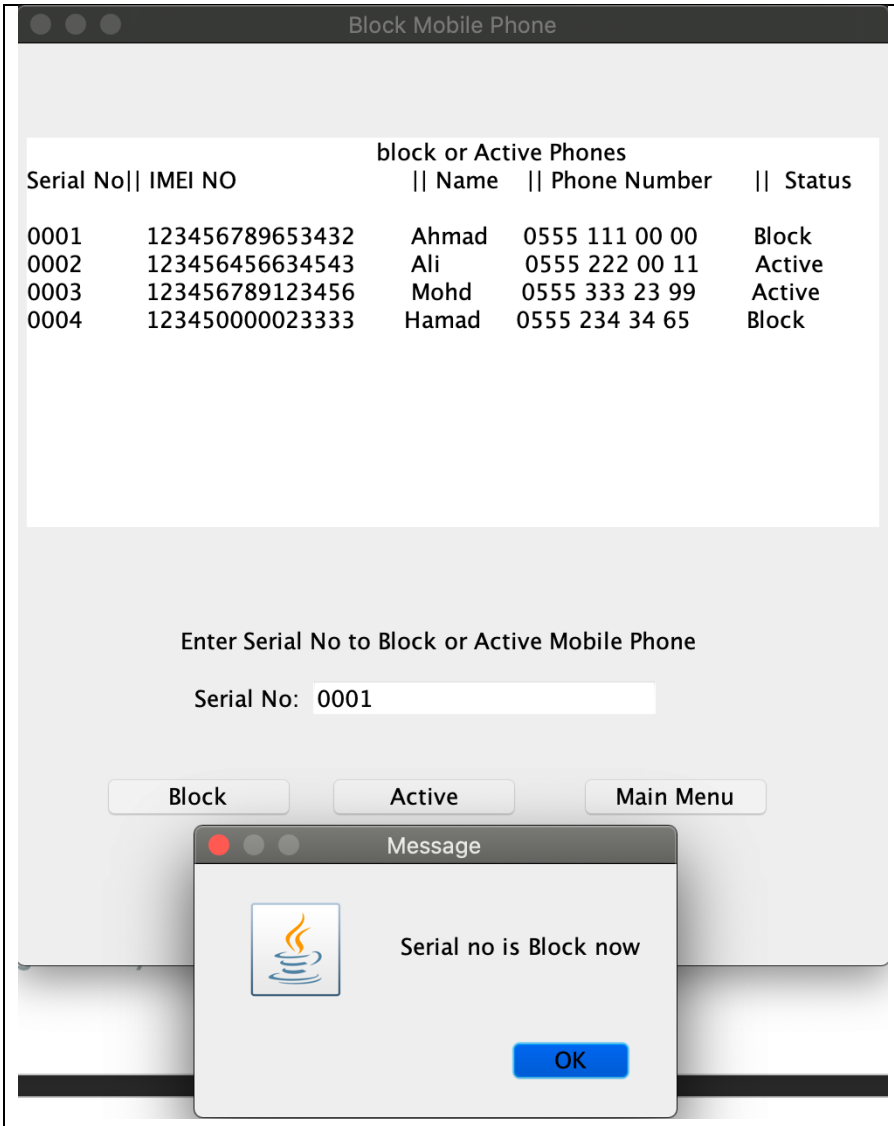


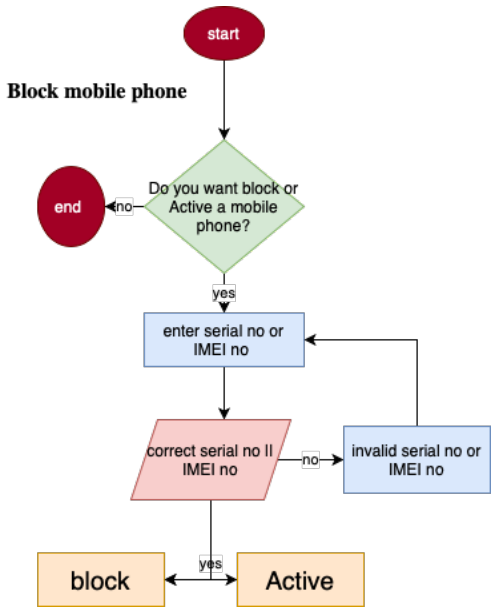




As shown in the figure to the side, The Track Mobile Phone page will help the user to keep track of his/her phone's live location using google map interface. The page will ask the user to input either Serial no. or IMEI no. and then it will go through the search engine and sees if the detail is valid or not. If valid/exist then It show the user the live location of the phone.

In this page, the system will use the double LinkedList source code in order to search for serial number or IMEI number in the database.

Block Mobile Phone page

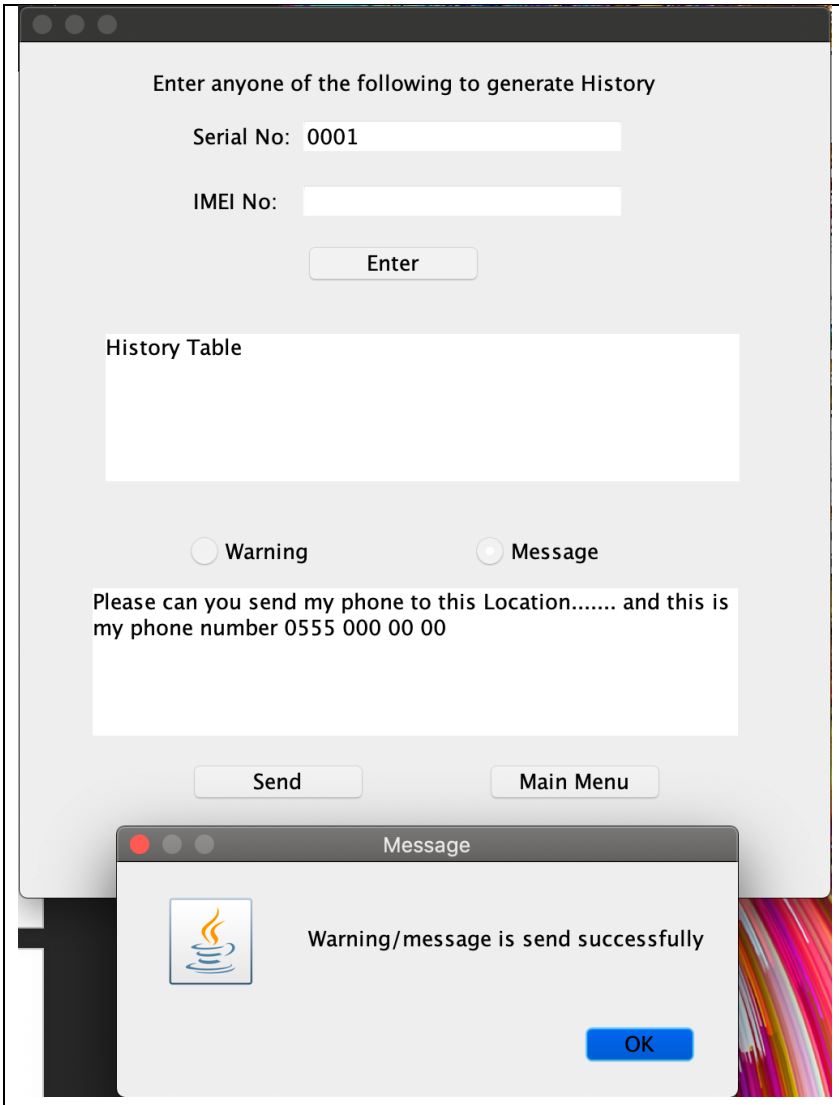


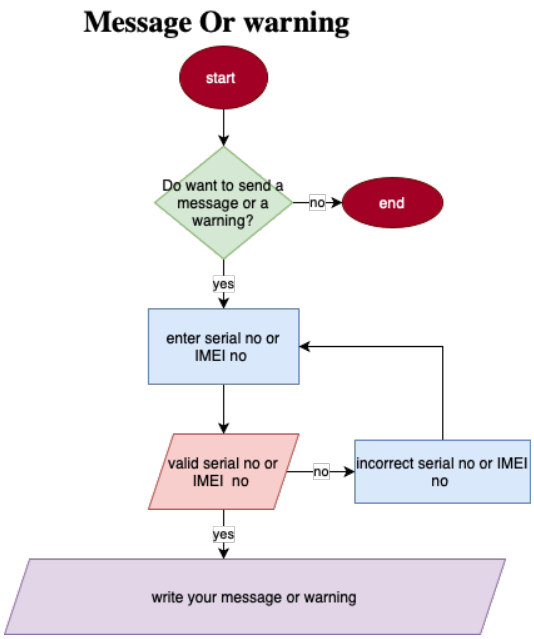




The Block Mobile phone page will help the user to see all the state of the registered phones and helps the user to block or active then. In this page the user is ask to enter the serial no of the phone which the user wishes to block or active it and it also has search engine to it so to determine whether the serial no exists in the system or not.

In this page, the system will use the double LinkedList source code in order to search for serial number in the database.

Send Warning page

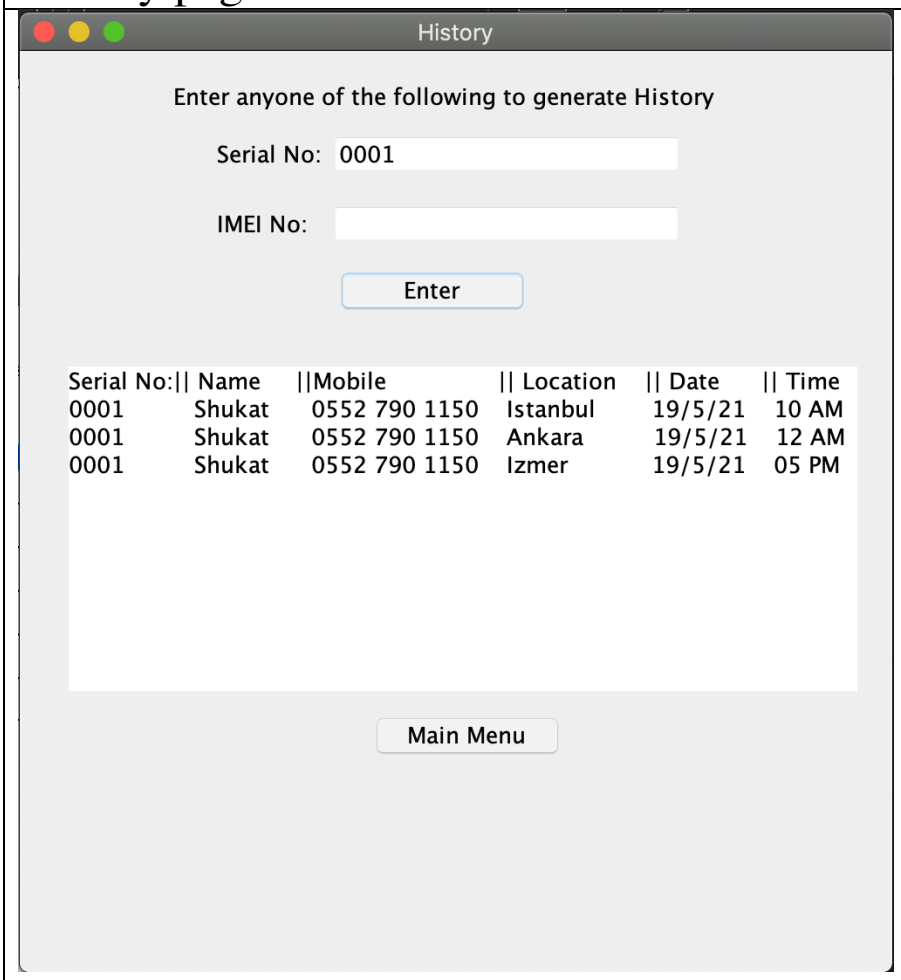



As shown in the figure to the side, The Send Waring page will help the user to send message or a warning to his/her phone's. The page will ask the user to input either Serial no. or IMEI no. and then it will go through the search engine and sees if the detail is valid or not. If valid/exist then the user can pick a warning or a message option and write a message in it and then send it to the person who has the phone at the moment. The page helps the user in case of lost phone or stolen phone and once the person received the message. He/She will turn the phone back to the user.

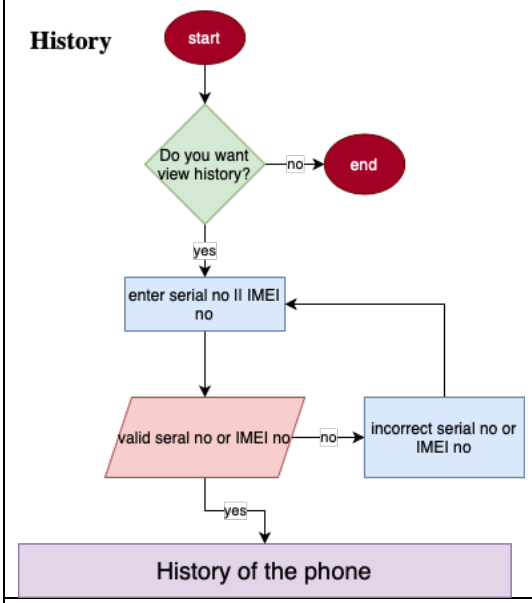
In this page, the system will use the double LinkedList source code in order to search for serial number or IMEI number in the database.

# History page




As shown in the figure to the side, The History page will help the user to keep track of his/her phone's all past history, location, date and time . The page will ask the user to input either Serial no. or IMEI no. and then it will go through the search engine and sees if the detail is valid or not. If valid/exist then the user can view the details. The page will help the user to know number about him/herself and place the visited the most during the period of time.

In this page, the system will use the double LinkedList source code in order to search for serial number or IMEI number in the database.




# Binary Tree Code for Register Mobile Phone Page

The binary tree class uses the node class which is a class that contains a key and a height. The BT class has some utility functions used by the main functions of the class. The 'height' function returns the height of a node that is passed to the function. The 'max' function returns the maximum height between two nodes. The 'getbalance' returns the balance factor of the tree. The 'preOrder' function basically prints out everything in the tree using the preorder traversal. The 'find' function finds a node in the tree.

Since the binary tree class uses AVL tree which is a self-balancing tree, the other utility functions are related to the insertion and deletion of a node in the tree. The 'leftRotate' and the 'rightRotate' functions are used to rotate the tree with respect to a node 'x' after insertion so that the tree will always have the right node at the right position so that the tree will maintain the O(log(n)) time complexity when deleting and updating a node in the tree. The 'minValueNode' function returns the node with minimum key value found in that tree and it is used in the 'deleteNode' function.

The 2 functions, 'insert' and 'deleteNode' are the main functions. The insert function starts with a normal binary tree insertion and then the height of the ancestor node is updated. After inserting a node, the function checks if the node is unbalanced. If the node is indeed unbalanced, there are 4 cases in which the function checks and determines what type of rotation will be applied. After that, everything is done. The 'deleteNode' function also starts with the standard binary tree deletion. After deletion, the height of the current node is updated and the balance factor of the node is checked. If it is unbalanced, there are 4 cases in which the function checks and determines what type of rotation will be applied.

```
package TrackingPhoneSystem;

public class BinaryTree {

     Node root;
    int height(Node N) {
        if (N == null)
            return 0;
        return N.height;
    }


    int max(int a, int b) {
        return (a > b) ? a : b;
    }

    Node rightRotate(Node y) {
        Node x = y.left;
        Node T2 = x.right;
        x.right = y;
        y.left = T2;
        y.height = max(height(y.left), height(y.right)) + 1;
        x.height = max(height(x.left), height(x.right)) + 1;
        return x;
    }


    Node leftRotate(Node x) {
```

```
        Node y = x.right;
        Node T2 = y.left;
        y.left = x;
        x.right = T2;
        x.height = max(height(x.left), height(x.right)) + 1;
        y.height = max(height(y.left), height(y.right)) + 1;
        return y;
    }

    int getBalance(Node N) {
        if (N == null)
            return 0;
        return height(N.left) - height(N.right);
    }

    Node insert(Node node, Information key) {

        if (node == null)
            return (new Node(key));
        if (key.getName().compareTo(node.key.getName()) < 0 )
            node.left = insert(node.left, key);
        else if (key.getName().compareTo(node.key.getName()) > 0 )
            node.right = insert(node.right, key);
        else
            return node;
        node.height = 1 + max(height(node.left),
                              height(node.right));

        int balance = getBalance(node);

        if (balance > 1 &&
key.getName().compareTo(node.left.key.getName()) < 0)
            return rightRotate(node);

        if (balance < -1 &&
key.getName().compareTo(node.right.key.getName()) > 0)
            return leftRotate(node);

        if (balance > 1 &&
key.getName().compareTo(node.left.key.getName()) > 0) {
            node.left = leftRotate(node.left);
            return rightRotate(node);
        }

        if (balance < -1 &&
key.getName().compareTo(node.right.key.getName()) < 0) {
            node.right = rightRotate(node.right);
            return leftRotate(node);
```

```
        }

        return node;
    }

    Node minValueNode(Node node)
    {
        Node current = node;

        while (current.left != null)
        current = current.left;
        return current;
    }

    Node deleteNode(Node root, Information key)
    {

        if (root == null)
            return root;
        if (key.getSerialNo().compareTo(root.key.getSerialNo()) < 0)
            root.left = deleteNode(root.left, key);
        else if (key.getSerialNo().compareTo(root.key.getSerialNo()) > 0)
            root.right = deleteNode(root.right, key);
        else
        {

            if ((root.left == null) || (root.right == null))
            {
                Node temp = null;
                if (temp == root.left)
                    temp = root.right;
                else
                    temp = root.left;

                if (temp == null)
                {
                    temp = root;
                    root = null;
                }
                else
                    root = temp;
            }
            else
            {


                Node temp = minValueNode(root.right);
                root.key = temp.key;
```

```
                root.right = deleteNode(root.right, temp.key);
            }
        }


        if (root == null)
            return root;
        root.height = max(height(root.left), height(root.right)) + 1;
        int balance = getBalance(root);
        if (balance > 1 && getBalance(root.left) >= 0)
            return rightRotate(root);
        if (balance > 1 && getBalance(root.left) < 0)
        {
            root.left = leftRotate(root.left);
            return rightRotate(root);
        }
        if (balance < -1 && getBalance(root.right) <= 0)
            return leftRotate(root);
        if (balance < -1 && getBalance(root.right) > 0)
        {
            root.right = rightRotate(root.right);
            return leftRotate(root);
        }

        return root;
    }


    void preOrder(Node node) {
        if (node != null) {
            System.out.print(node.key.getName() + " ");
            preOrder(node.left);
            preOrder(node.right);
        }
    }


    Information find(Node node, String serialNo) {
        if (node != null) {
            if(node.key.getSerialNo() == serialNo) {
               return node.key;
            }
            preOrder(node.left);
            preOrder(node.right);
        }
        return null;
    }
}
```

# Double Linked List Source code for Search Engine

There are 3 variables in the node. One of them is also reserved for data. The other two variables are pointers, one holding the node before it and the other holding the next node.
The first node of the list has its previous link pointing to NULL similarly the last node of the list has its next node pointing to NULL.
Adding Node
The previous pointer of first node will always be NULL and next will point to front. If the node is inserted is the first node of the list, then we make front and end point to this node. Else we only make front point to this node.
Search Node
The number to be found is written to the search node method. Each node is checked one by one to find out if there is a match.
If there is a match, the checking process is terminated, and output is given through the console.

```
package TrackingPhoneSystem;

import TrackingPhoneSystem.Node;

public class DoubliLinkedList {
    Node first, last = null;

    public void addNode(int detail) {

        Node newNode = new Node(detail);
        if(first == null) {

            first = last = newNode;
            first.previous = null;
            last.next = null;
        }
        else {
            last.next = newNode;
            newNode.previous = last;
            last = newNode;
            last.next = null;
        }
    }
    public void searchNode(int value) {
        int k = 1234567895;
        boolean serialno = false;
        Node current = first;
        if(last == null) {
            System.out.println("there are no phone ");
            return;
        }
        while(current != null) {
            if(current.detail == value) {
```

```
                serialno = true;
                break;
            }
            current = current.next;
            k++;
        }
        if(serialno)
             System.out.println(" Date found in the System " + k);
        else
             System.out.println("Invlid Serial Number or IMEI Number");
    }

}
```

## Source Code

The source code will be attached in zip file and will be uploaded separately.